

CHAPTER 7

INPUT/OUTPUT (I/O) AND INTERFACING

INTRODUCTION

The input/output section, under the control of the CPU's control section, allows the computer to communicate with and/or control other computers, peripheral devices, other subsystems (display and communication), and systems (fire control, sonar, FTAS, and the like). Take the time to understand your computer's I/O section: its organization, operation, and interfacing format. The latter is very important because if the interfaces for the computer and the external equipment do not match, your computer will not talk to anyone.

After completing this chapter you should be able to:

- Understand the terminology associated with I/O
- Describe how your computer's input/output is organized—hardware and software
- List and describe how the different operating modes affect the transfer of information
- Describe the circuits and their functions in I/O operations
- Describe the categories of I/O operations
- Recognize how the external equipment in your computer's system is connected
- List and describe the types of interfaces used in I/O operations
- Describe serial data I/O operations
- Describe parallel data I/O operations

Let's begin your study of input/output with how it is organized in your computer. The different types of computers vary in their organization of I/O, but the basic operations of the serial and parallel **interfaces** are similar regardless of the computer types.

TOPIC 1—TERMINOLOGY

You should be familiar with the following terms before studying this chapter:

- ANEW —Army-Navy Electronic Warfare.

- ANSI —American National Standards Institute.

- Data Communications Equipment (DCE) —Any device that communicates the data; for example, a modem.

- **Data Terminal Equipment (DTE)** —Any device that can transmit or send data; for example, a computer.

- **EIA** —Electronics Industry Association.

- **External Function (EF) data** —The purpose of the EF function is to transfer command information by using the appropriate control signals from the transmitting computer to the receiving device. The word size and bit format of the EF data will be specified by the appropriate system design data or the individual equipment specifications.

- **External Interrupt (EI) data** —The purpose of the EI function is to transfer status information by using the appropriate control signals from a transmitting device to the receiving computer. The word size and bit format of the EI data will be specified by the appropriate system design data or the individual equipment specifications.

- **Gateway** —A device that serves as a shared entry point from a local area network into a larger information resource such as a mainframe computer.

- **Handshaking** —Signals necessary for completing I/O operations.

- **Hub** —Repeats the signal on the cable.

- **IEEE** —Institute for Electrical and Electronics Engineers.

- **Input** —Input refers to input to the computer.

- **Input/Output (I/O) word** —The I/O word is defined as a digital word of a specified number of bits, which has been agreed upon as the basic unit of communication between interconnected units.

- **Input Data (ID)** —The purpose of the ID function is to receive information using the appropriate control signals from a transmitting device by the receiving computer. The word size and bit format of the ID data will be specified by the appropriate system design data or the individual equipment specifications.

- **IOA** —Input/output adapter.

- **IOC** —Input/output controller.

- **Output** —Output refers to output from the computer.

- **Output Data (OD)** —The purpose of the OD function is to transfer information using the appropriate control signals from a transmitting computer to the receiving device. The word size and bit format of the

OD will be specified by the appropriate system design data or the individual equipment specifications.

- **Protocol** —In a computer, protocol is the procedure required to initiate and maintain operations. For example, I/O operations of a parallel format use a **request** and an **acknowledge** protocol to perform input and/or output operations for the transfer of information between the computer and external equipment.

- **RS** —Recommended Standard.

- **Sink** —The sink is defined as that end of a channel that receives information frames.

- **Source** —The end of a channel that transmits information frames.

TOPIC 2—INPUT/OUTPUT (I/O) ORGANIZATION

All computers are capable of I/O operations. Some computers rely on the CPU to handle all operations including the I/O operations. These computers simply use the circuits in the CPU to handle the I/O operations. However, the majority of computers use an I/O processor (fig. 7-1) that enhances the capabilities of the computer and relieves the burden of I/O processing from being on the CPU. This allows the computer to perform other operations while still performing I/O operations. In this topic we discuss I/O operations in general terms, using an I/O processor. This includes the physical aspects, data arrangement, format, instructions, operations (modes of operation, timing,

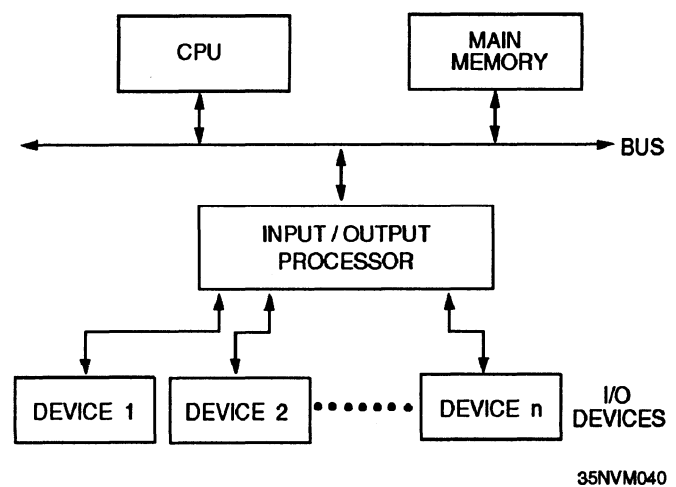


Figure 7-1.—I/O processor in a computer system.

and control), categories of I/O operations, and I/O interfacing.

INPUT/OUTPUT PROCESSOR

For those computers that have an I/O processor, the physical organization of I/O is similar to the other major functional areas: CPU and memory. I/O processors can vary from many pcb's that makeup a module/unit to a single pcb. Larger mainframe computers use the modular arrangement: multiple components on multiple pcb's that comprise one or more modules or units. Mini- and microcomputers use chassis or assemblies, cages or racks, and motherboard/backplane arrangements. Minis and micros use multiple components on one pcb or groups of pcb's (usually not more than seven) to form the I/O processor.

The I/O processor controls the transfer of information between the computer's main memory and the external equipments. I/O processors are packaged two different ways: (1) IOC/IOA modules or multiple IOC/IOA pcb's, and (2) I/O pcb's. Regardless of the setup, computers with an I/O processor will use some sort of controller to regulate the signals in the I/O processor itself (includes IOC/IOA setup) and memory.

IOC/IOA Module or Multiple IOC/IOA Pcb's

I/O processors that are packaged as IOC/IOA modules or multiple IOC/IOA pcb's are divided into two sections. The two sections are a single module/unit or group of pcb's for the **I/O controller (IOC)** and a single module/unit or group of pcb's for the **I/O adapter (IOA)** (fig. 7-2). Mainframes and some minis use this arrangement.

IOC.— The IOC relieves the CPU of the necessity to perform the time consuming functions of establishing, directing, and monitoring transfers with external equipments. Data and control signals are exchanged with external equipments via the IOA. IOCs communicate by means of a bidirectional bus. An IOC is provided with a repertoire of instructions (commands) that varies with the type of computer. The IOC contains the necessary control and timing circuits (digital) necessary to function asynchronously with the CPU and controls the transfer of data between accessible main memory and the external equipments. IOC programs are initiated by instructions from the CPU and executed by a repertoire of IOC commands stored in main memory. Included in the repertoire are those commands that establish the conditions for data

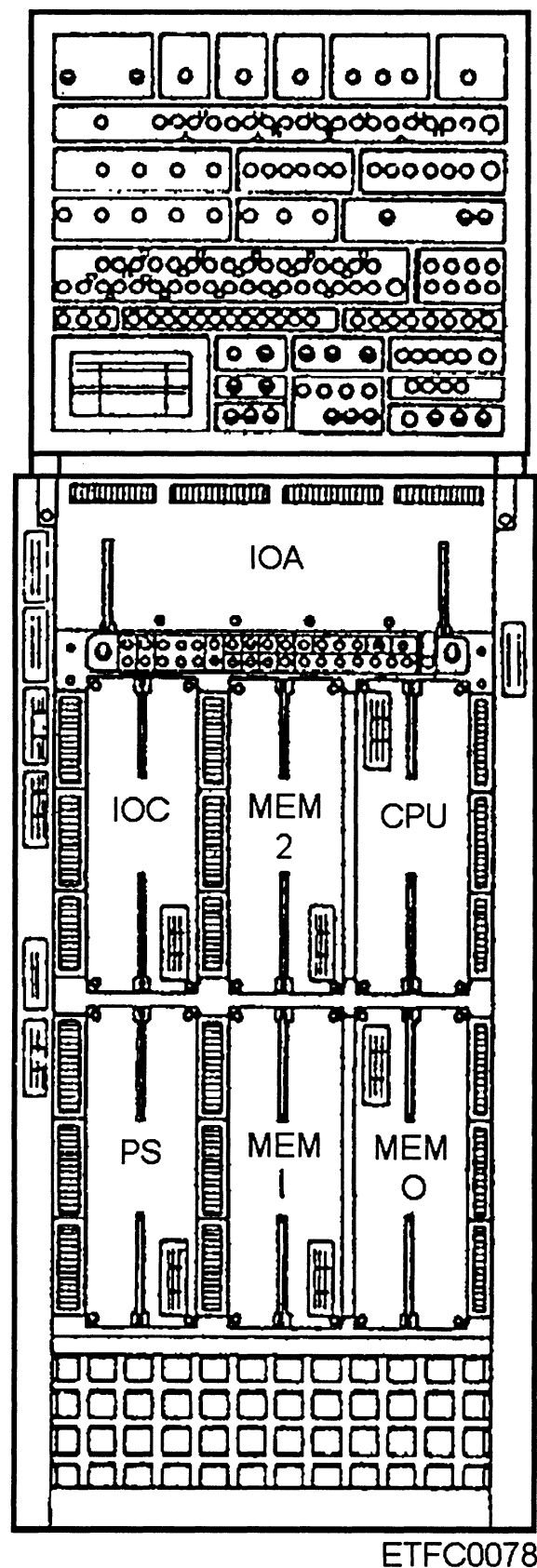


Figure 7-2.—IOC/IOA modules in a single cabinet configuration.

transfers to and from the external equipments. See figure 7-3 for an example.

IOA.— The IOA changes the input and output control and data signal voltages to the voltage requirements of the computer or external equipments. The IOA receives data and control signals from the IOC logic of the computer, and returns data and interfacing signals to the IOC logic. It also transfers data and control signals to the external equipments and receives data and interfacing signals from the external equipments. The IOA logic circuits consist primarily of line drivers/receivers (linear circuits) and timing circuits (digital circuits).

Communication between the IOC and IOA is by means of a bidirectional bus. The IOA communicates

with the external equipments via I/O channels/ports. The connectors for the input and output channels or ports are physically located atop the IOA unit (fig. 7-4) or on the rear of a computer cabinet (fig. 7-5). The type of interfacing will dictate the type of connectors for the channels or ports. The IOA is capable of receiving and sending parallel and serial data.

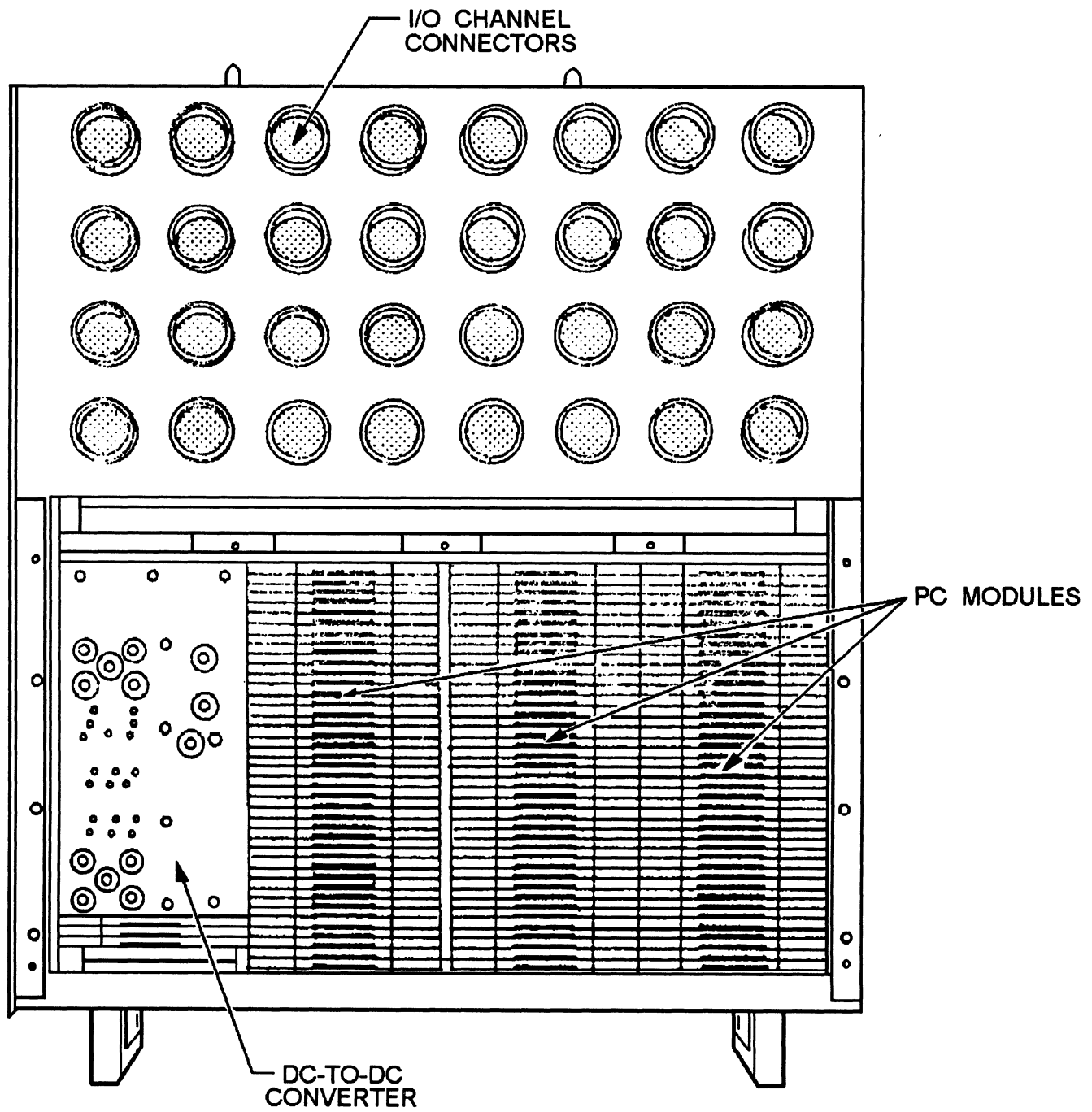
IOC/IOA INTERFACING.— The IOA is a completely passive unit and functions under the direct control of the IOC. The driver circuits pass interfacing and data signals to the external equipments. The receivers pass data to the IOC. They are directed by the IOC using input and output control circuits. The request circuits pass interface signals to the IOC

I/O CONTROLLER (IOC) INSTRUCTIONS

| OCTAL FORMAT | CODING FORMAT | | NAME | DESCRIPTION | CC/CD | F | ISA REF | |
|-----------------|------------------|--------|--------------------------------------------------------------|-----------------------------------------------|------------------------------------------|-----|------------|------|
| 14 k = 2 | TFB | j,c,m | Terminate External Function Buffer and Chain Activity on Cj | Terminated EF | m = 1 Allow Queued Monitor Interrupts | N | I | B.9 |
| 14 k = 3 | TXB | j,c,m | Terminate External Interrupt Buffer and Chain Activity on Cj | Terminated EI | | N | I | B.9 |
| 15 k = 0 | IMIR | j,c | Generate Input Monitor Interrupt Request on Cj | Generate input monitor interrupt on Cj | | N | I | B.10 |
| 15 k = 1 | OMIR | j,c | Generate Output Monitor Interrupt Request on Cj | Generate output monitor interrupt on Cj | | N | I | B.10 |
| 15 k = 2 | FMIR | j,c | Generate External Function Monitor Interrupt Request on Cj | Generate EF monitor interrupt on Cj | | N | I | B.10 |
| 15 k = 3 | XMIR | j,c | Generate External Interrupt Monitor Interrupt Request on Cj | Generate EI monitor interrupt on Cj | | N | I | B.10 |
| 16 k = 0 | AIC | j,y,c | Set Input Chain Active on Cj | y → CAP _{kj} activate chain on Cj | | N | I | B.11 |
| 16 k = 1 | AOC | j,y,c | Set Output Chain Active on Cj | | | N | I | B.11 |
| 16 k = 2 | AFC | j,y,c | Set External Function Chain Active on Cj | | | N | I | B.11 |
| 16 k = 3 | AXC | j,y,c | Set External Interrupt Chain Active on Cj | | | N | I | B.11 |
| 17 m = 0 | TBZ | kj,y | Test External Interrupt Chain Active on Cj | If (Y) _{kj} = 0, skip NI; else NI | | Y | I | B.12 |
| 17 m = 1 | TBS | kj,y | Test Bit Zero | If (Y) _{kj} = 1, skip NI; else NI | | Y | I | B.12 |
| 17 m = 1 | TBS | kj,y | Test BIT Set | y → CAR or CAP; Jump to Y | | N | I | B.13 |
| 20 | JIO | y,c | Jump to Y | N/A | | N/A | | |
| 21 | N/A | | Illegal | (Y) → IOC CMA _{mkj} | | N | I | B.14 |
| 22 | LICM | kj,y,c | Load IOC Control Memory | (Y) → IOC RTC | | N | I | B.15 |
| 23 | ILTC | y,c | Load Realtime Clock | (IOC CMA) _{mkj} → Y | | N | I | B.16 |
| 24 | SICM | kj,y,c | Store IOC Control Memory | 1 → (Y) _{kj} | | N | I | B.17 |
| 25 (Bit 25 = 0) | IBS | kj,y,c | Set Bit | | | N | I | |

ETFC0079

Figure 7-3.—Example of a repertoire of IOC commands.



ETFC0080

Figure 7-4.—IOA, top view with I/O connectors.

I/O CONNECTORS

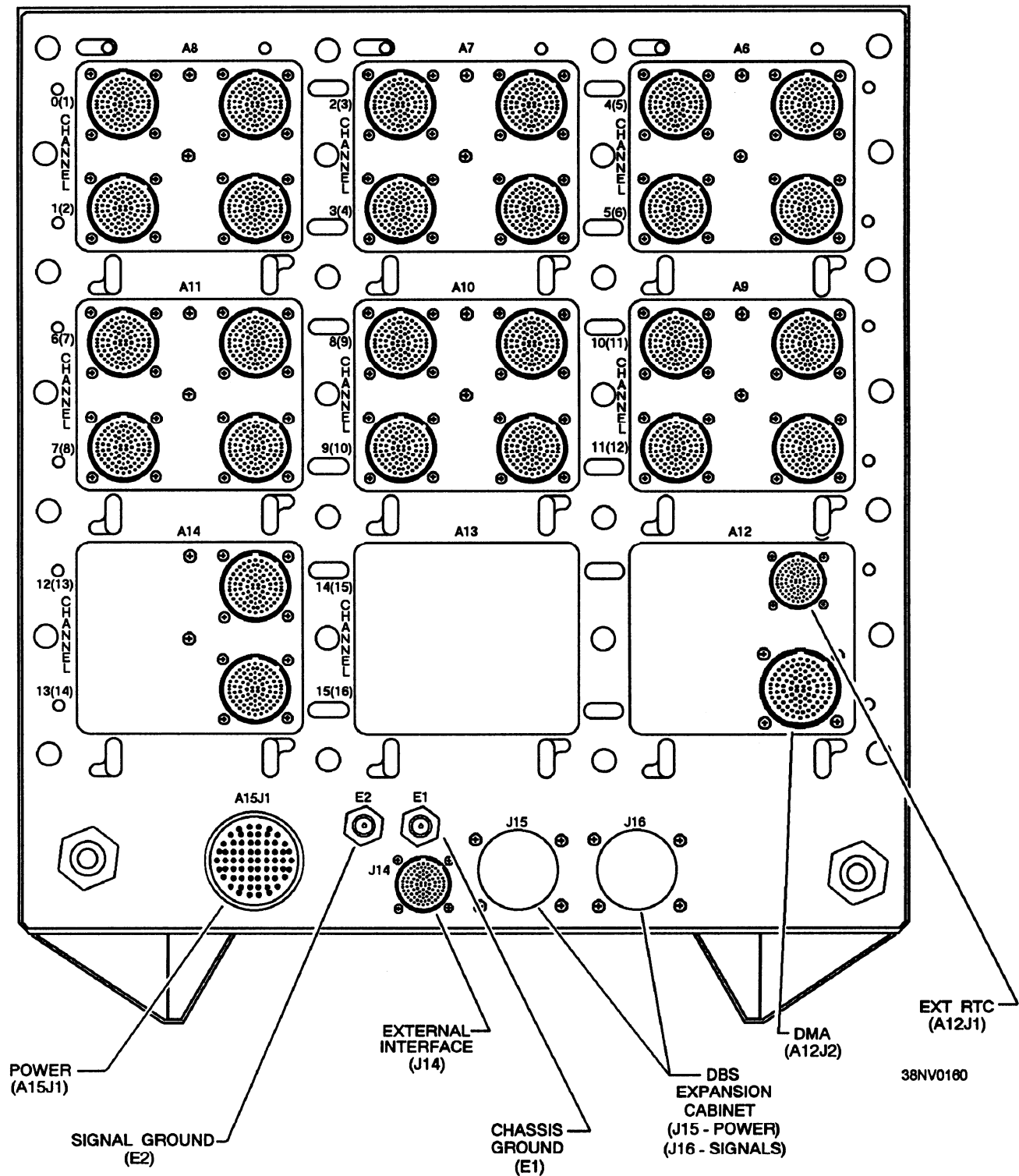


Figure 7-5.—I/O connectors, rear of computer cabinet.

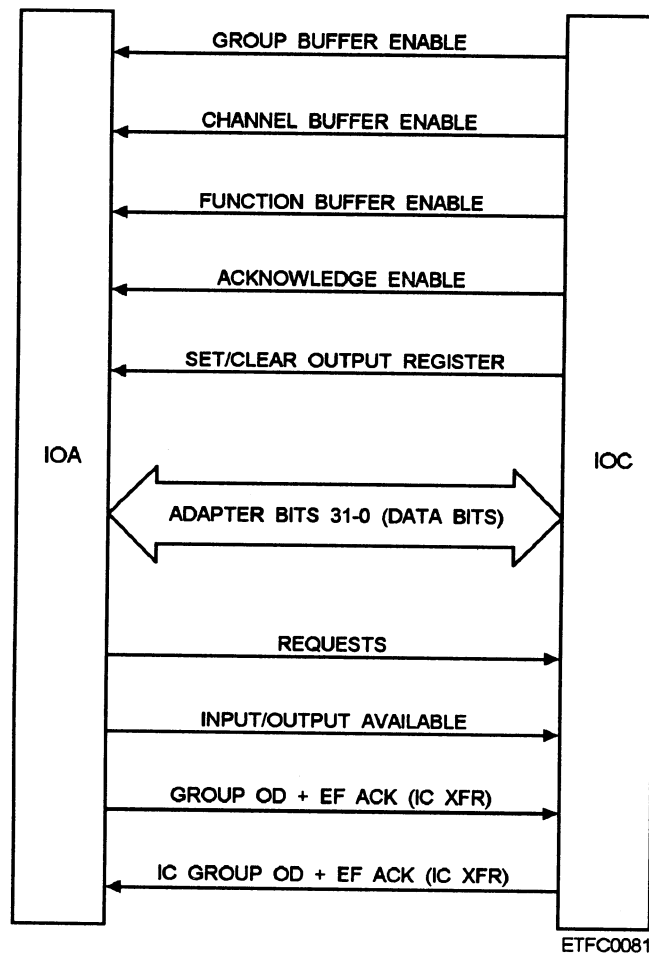


Figure 7-6.—IOA/IOC interface.

(fig. 7-6). Some of the data and control signals exchanged between the IOC and IOA include: “

- Buffer enables
- Acknowledge enables
- Set/clear output register
- Data bits
- Request lines
- Input/output available

I/O Pcb(s)

In the I/O pcb arrangement, minis and micros have multiple I/O pcb's or a single I/O pcb. When multiple I/O pcb's are used, each I/O pcb will be assigned a number of external equipments for I/O operations. In this arrangement other circuitry will be used that basically performs the same duties as a controller. In the single I/O pcb arrangement, the functions that an IOA would perform are contained on

the pcb to match the electrical **interface** of the external device(s) to that of the computer. The connectors for the input and output channels/ports are usually located on the rear of the I/O pcb (fig. 7-7).

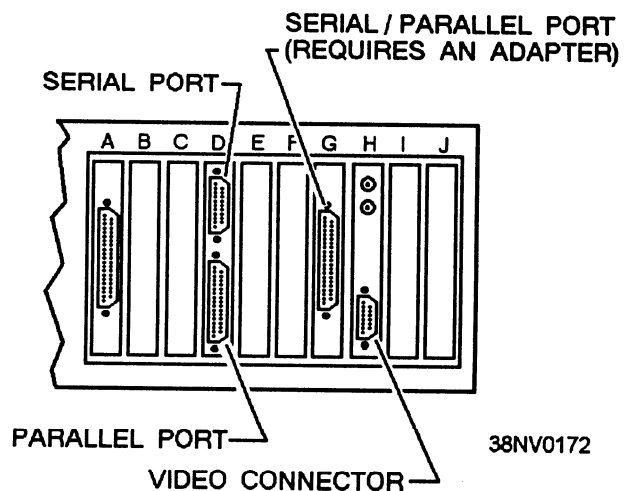


Figure 7-7.—Connector parts on the rear of a microcomputer.

Some arrangements include assigning multiple ports to each channel (fig. 7-8).

Micros usually have only one pcb for their I/O operations: the pcb has both a parallel and a serial port (fig. 7-7). Some minis and micros have dedicated pcb's separate from the I/O pcb(s) to handle the interface for the peripherals and displays. For micros, the interfacing for the keyboard is usually located on the I/O pcb.

INPUT/OUTPUT DATA ARRANGEMENT

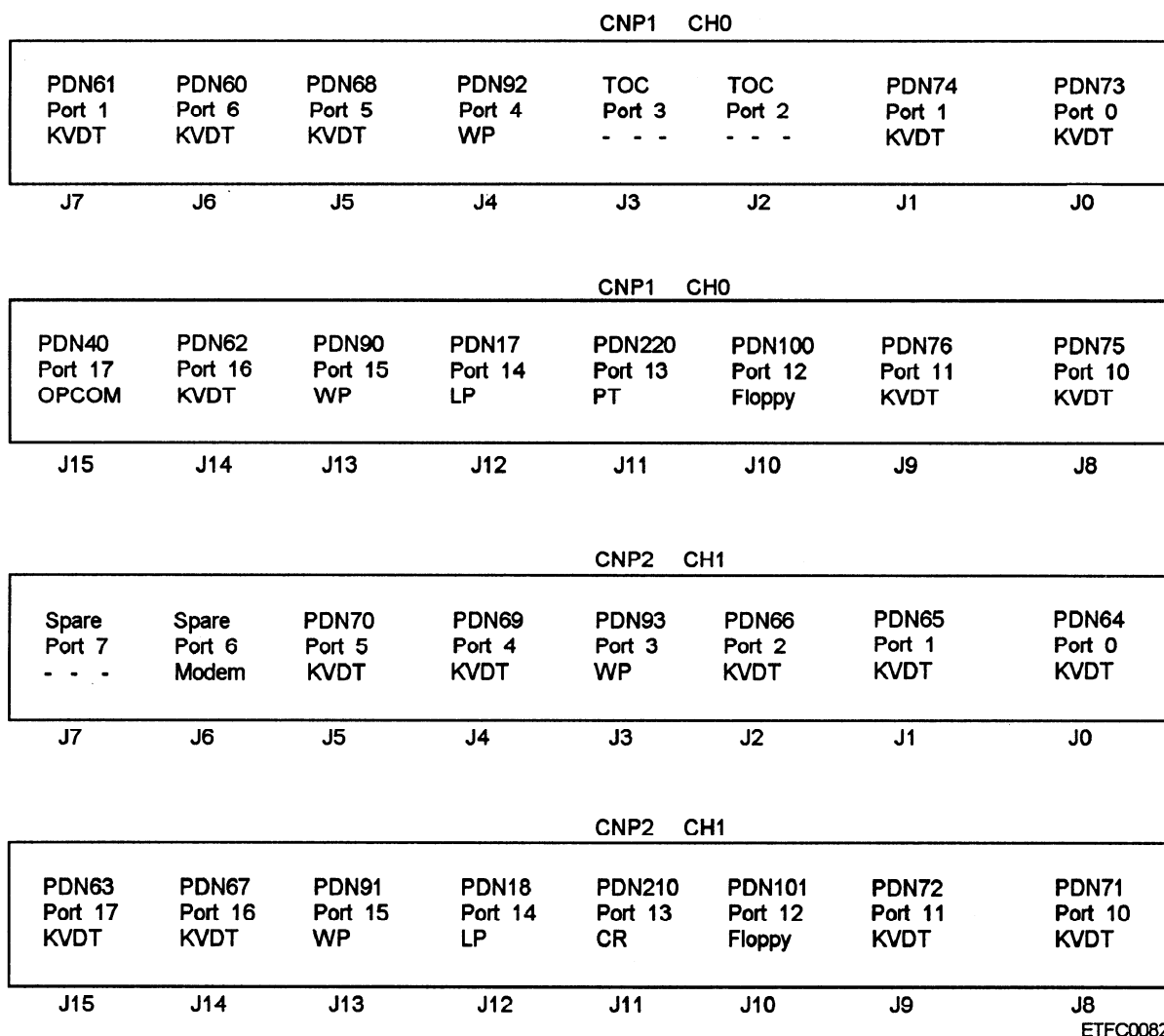
The function of any I/O operation is to exchange information between equipments. Regardless of the techniques used to move the information, there are consistencies in the architectures of the I/O sections used by computers. These consistencies include the **arrangement** of the information exchanged and the **format** of the information exchanged.

Arrangement

The types of information exchanged between the computer and the external equipments fall into two basic categories: **data words** and **control words**. The length of the information exchanged varies with the type of computer from 8-bit words to 32-bit words.

DATA WORDS.— Data words represent the alphabetic and numeric information exchanged. Data words are always thought of with the computer as the reference point. Input data words are data entered in the computer from equipments external to the computer. Output data words are data sent **out** to the external equipments from the computer. Some computers transfer data words that include data and externally specified addresses and index addresses.

CONTROL WORDS.— Control words specify an action to be accomplished by an external equipment. This might include an error or special condition of an



ETFC0082

Figure 7-8.—Assigning multiple ports to a single channel.

external equipment or the status of an external equipment, in response to a computer control word. Some examples of control words used by computers include the following:

- Function (command) control words —Function control words are sent by the computer to an external equipment to specify the type of operation it is to perform. The signals used for the control words are often referred to as **handshaking**. An example of a control word would be a function code word telling a printer to print the contents of a specific accumulator register at the location specified by the address in the instruction. Computers that have a control memory use a control memory word to transfer data for I/O buffer operations.

- External interrupt words —External interrupt words are sent to the computer to specify that an error or special condition exists in an external equipment or the status of an external equipment. Review chapter 5 of this volume for a detailed discussion of interrupts: their classification, types (micro, mini, and mainframe), priorities (micro, mini, and mainframe), codes, and handling process.

Format

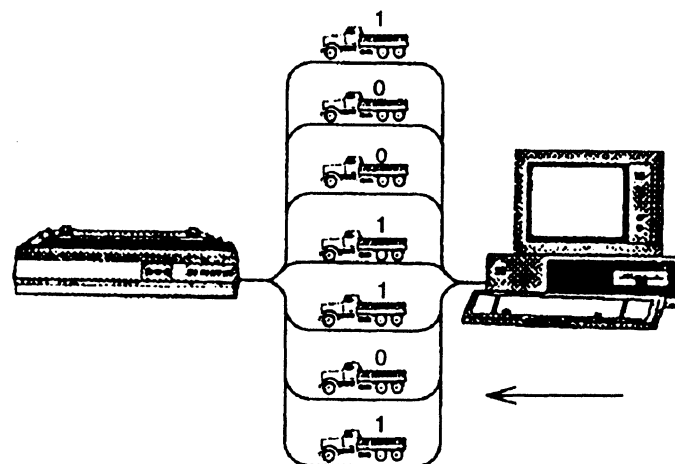
There are two formats of information exchanged by a computer: **parallel** and **serial**. The type of interface will dictate the format of the information exchanged.

PARALLEL.— When the computer exchanges information using a parallel configuration, all bits of information represented by a byte or word are input or output simultaneously. In figure 7-9, frame A, we illustrate how the character M is output from the computer to a printer in parallel format.

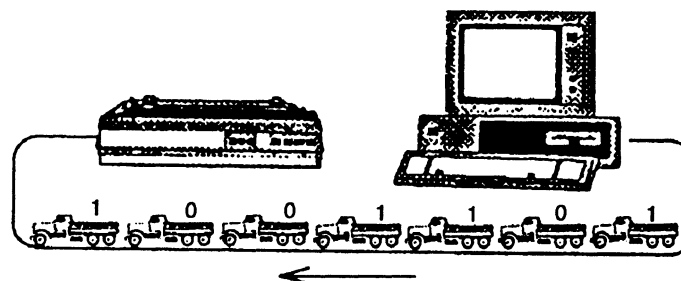
SERIAL.— When the computer exchanges information using a serial configuration, all bits of information are input or output one at a time. Figure 7-9, frame B, illustrates the character M being output from the computer to a printer in serial format.

INPUT/OUTPUT INSTRUCTIONS

The heart of the I/O section is the input/output processor: an IOC/IOA or I/O pcb arrangement. All computers have I/O instructions. Computers without an IOC/IOA arrangement have other means of



A. M TRANSMITTED IN PARALLEL



B. M TRANSMITTED SERIALLY

35NVM029

Figure 7-9.—Parallel and serial configurations: A. Character M transmitted in parallel; B. Character M transmitted serially,

optimizing the CPU's time, so the CPU is not involved in all transactions, including the I/O instructions. We cover those methods later in this topic. However, for computers that have an IOC, the IOC is a processor in its own right. We focus our discussion of I/O instructions on I/O processors with an IOC. An IOC is capable of executing its own set of instructions specifically designed to govern I/O operations for those channels/ports handled by the particular IOC. Figure 7-10 shows the format of an example IOC instruction. This format is used for some mainframes and some minis. The designators shown are for a typical I/O instruction and may vary with IOC instructions. Each IOC executes instructions stored in main memory in the same manner as the CPU executes instructions. There are two basic types of IOC instructions: **command instructions** and **chaining instructions**.

Command instructions are executed by the IOC under the control of the CPU's main program. Chaining instructions are executed under the command of an active channel (I/O operation in progress) chain. Some IOC instructions perform the same functions whether it is a command or a chaining instruction.

Command Instructions

Command instructions provide control over IOC single or dual channel operations. They are executed individually using the following process. The CPU executes an **I/O command start instruction**, which is a CPU instruction. The I/O command start instruction specifies or addresses an IOC(s) and then halts further CPU processing. The addressed IOC then references

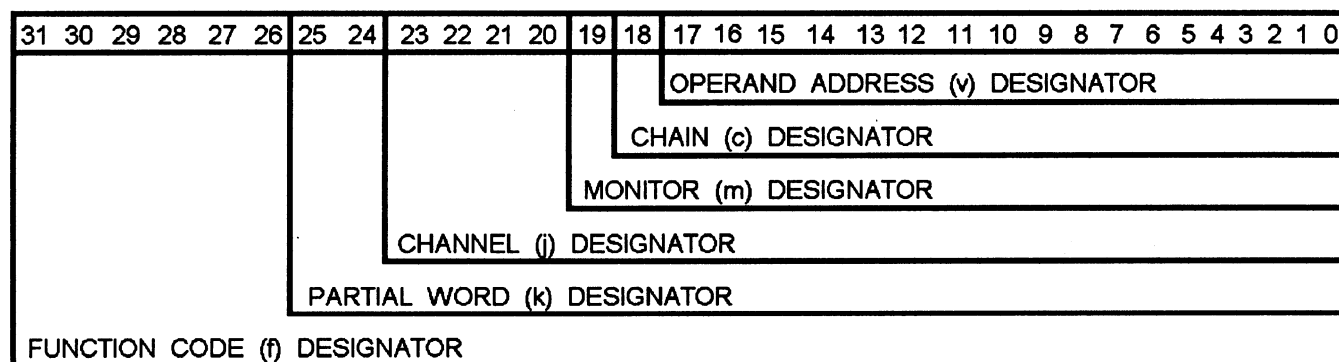
specific main memory addresses (the command cell) and executes the IOC command instruction previously stored in the addresses. At completion of the instruction execution, the IOC will clear assigned bits of the command cells to indicate to the CPU that the command has been processed and to release the CPU to continue further processing. This is one instance in interrupt driven I/O operations where the CPU will delay processing while waiting for an I/O operation to take place.

The instruction contained in the command cell will cause the IOC to perform a variety of channel activity functions. The most common operations deal with initiating a new chain or terminating a chain in progress. Other commands are used to master clear individual channels, enable or disable a variety of interrupts, monitor channel status, load or store control memory, and initiate the IOC built-in test (BIT).

Chaining Instructions

Chains of IOC instructions are stored in memory by the main CPU program before the I/O operation takes place. The actual execution of chaining instructions is independent of the CPU. Only a command instruction execution from the command cell will delay CPU processing. There can be an input chain and/or an output chain being executed for each channel. Input or output chains deal primarily with the transfer of blocks of information.

A chain consists of IOC control words, command words, output data words, and specified locations for



38NV0162

Figure 7-10.—IOC instruction format.

external status words and data words returned (input) from the channel. The starting address of the chain (and other data) is provided by the load control memory command. The chain's starting address is stored in the channel's **chain address pointer** portion of I/O control memory. The contents of the I/O control memory are used by the IOC to control all channel operations including execution of chaining instructions.

TOPIC 3—INPUT/OUTPUT OPERATIONS

Input/output operations are initiated by the CPU. Computers with an IOC will begin I/O control functions only after an initiate I/O or equivalent instruction is executed by the CPU. I/O operations under the control of the computer program control the external equipment. Computer instructions inform the external equipment which type of operations to perform with function codes. Computer instructions also specify memory areas for input and output information. Input/output operations do not accept data from external equipments or send information to them unless memory areas for the data have been specified by the computer programs. Whenever an external equipment is ready to send or receive data, a request signal is sent to the computer. How the I/O section notifies or interrupts the control section that an external equipment is ready to send or receive information/data depends on the type of computer. Some constants in all I/O operations include the following:

- When the transfer will begin,
- How many words or bytes will be transferred,
- Word or byte size,
- When each individual word or byte is actually transferred, and,
- When the transfer will terminate.

I/O operations require circuitry that must take action in a specific sequence of events to communicate with the external equipment. In I/O operations, we examine operating modes, I/O circuits, and I/O functions.

OPERATING MODES

Similar to the CPU, some computers have the capability to select operating modes. These options are usually found with computers that have an IOC. They can be found on the computer's controlling device,

usually a maintenance panel or some equivalent. You can use this option for troubleshooting purposes. Consult the operator's section of your computer's technical manual. As far as the operating modes for I/O operations, these options are usually established at the factory. Again, they usually apply to computers that have an IOC. Some of the operating modes for I/O operations include the following:

- Single-channel —The single-channel operating mode allows external equipments to communicate with the computer via one input/output channel.

- Dual-channel —The dual-channel operating mode is used by computers with smaller word sizes, say 16 bits, to communicate with external devices using a larger word size (30 or 32 bits). In a dual/channel mode, the data lines for two channels are combined under control of the lower order channel. A pair of sequentially numbered channels (0 and 1, 2 and 3, and so forth) is used for dual-channel operations. The even numbered channel provides the control signals and lower half or lower order data bits. The odd numbered channel provides the upper half or upper order data bits only. The exchange of information over the dual channel is controlled by the even numbered channel's interface signals. Dual channels may use the computer peripheral or intercomputer channel signals.

- Externally specified address (ESA) —The externally specified address mode provides the external devices with a means of specifying an absolute memory location for storage (write) or retrieval (read) of information on a word-by-word basis.

- Externally specified index (ESI) —The externally specified index mode is identical to regular transfers (input, output, external interrupt, and external function) except that the IOC requires the external device to specify an index address in main memory.

- Intercomputer channel (IC) —The intercomputer channel mode permits communication between two CPUs. In this mode, each computer appears as an external device to the other. During operations, the computer that is outputting the data is defined as the sending computer. The computer that is receiving the data from the sending computer is defined as the receiving computer.

I/O CIRCUITS

In chapter 4, we discussed the circuits used by computers. We also discussed some of the same circuit types in the CPU and memory sections. I/O is no

different; but in addition, I/O operations include not only digital ICs, but also linear ICs. The linear IC circuits are the first and last type of circuitry the information interfaces with when entering and leaving the computer. In this topic, we discuss some of the more common circuits you will encounter when dealing with I/O functions. In addition to the circuits we have discussed in the CPU and memory sections, you must be familiar with **driver** and **receiver** circuits (linear ICs). Review chapter 4 of this volume and NEETS, Module 13, *Introduction to Number Systems and Logic Circuits*. They provide excellent reviews of the circuits and their functions covered in the remainder of this topic. The circuits include:

- Adders
- Command signals (enables)
- Decoders
- Line drivers and receivers
- Registers (includes RTC and Monitor Clock for IOCs)
- Selectors
- Timing
- Translators

One of the primary uses of registers in I/O operations is to provide the interfacing between the CPU, I/O, and memory. They enable and route **control** and **data** information between the CPU, I/O, and memory using the internal bus system. In a computer with no I/O processor, a register will be designated as either an input or output register (fig. 7-11). Decoder circuits are used for address translation, control circuits for governing the operation of the interface, data registers, and status registers for information exchange. The data registers are used to hold or **buffer** data during interchanges between the very fast CPU and the slower external equipments. The status registers hold information for the CPU that indicates the operating condition and current activities of the external equipments. We discuss external interfacing later in this topic.

INPUT/OUTPUT FUNCTIONS

The input and output functions performed by an I/O processor are defined and enabled through the interpretation and execution of input/output and/or input/output controller (I/O(C)) commands obtained from main memory. The I/O circuits provide the

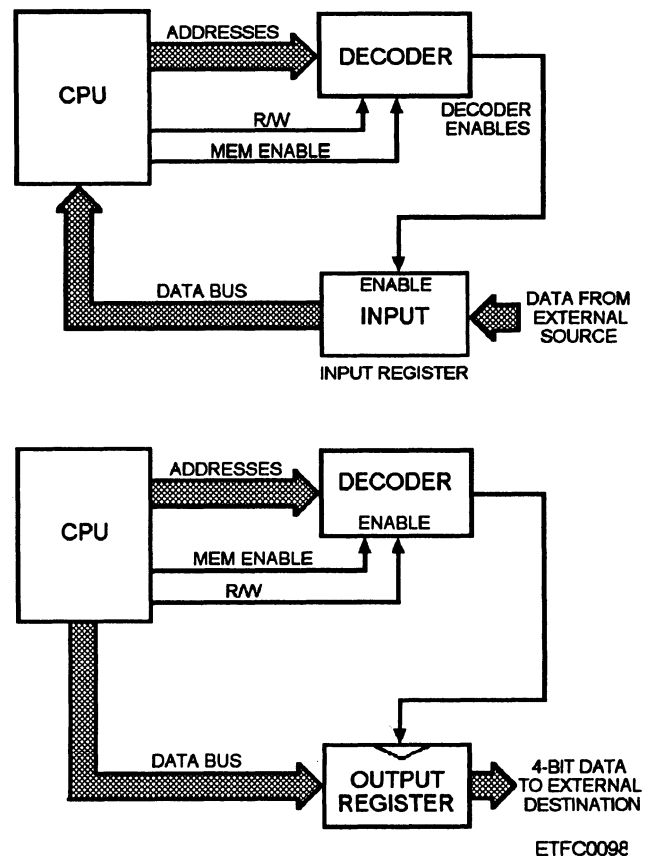


Figure 7-11.—Computer operations with no I/O processor using registers.

timing, control, temporary storage, routing, command translation, and interfacing (internal and external) to perform I/O operations.

Timing Circuits

As we discussed in the memory section, timing circuits also will provide the enables to manage the I/O control circuits used for I/O operations. Some computers use the computer's master clock and one or two other timing signals derived from the master clock to control the flow of data in I/O operations; an example of this is the timing used in microcomputers. Still other more complex computers, such as mainframes and minicomputers, rely on a master clock and main timing circuits in the respective functional area (CPU, memory, or I/O) to produce and distribute timing signals to the I/O control circuits.

In computers with an IOC, their I/O master clock and timing circuits operate completely independently of the CPU timing. Their master clock is started when the computer is initially powered on or auto restarted. It can only be stopped or temporarily halted under

certain conditions, such as a computer master clear or a read/write memory reference.

Control Circuits

The I/O control circuits are under the direct or indirect control of the program. The I/O control circuits decode I/O commands from the CPU and generate the required signals to execute the instructions. The timing circuits coordinate control circuit operations. Computers with an IOC operate independently from the CPU after they receive an initiate I/O instruction and control all I/O operations. Depending on the computer type, some of the more common uses of the control circuits include the following:

- Logic to decode I/O commands
- Logic to generate signals to execute I/O instructions
- Logic to evaluate priorities of I/O requests
- Logic to execute buffered and unbuffered requests

A term used quite often with I/O control operations is the term *buffer*. A buffer is nothing more than a sequential set of memory locations that contains data to be sent out or an area that is set aside for data to be received. A buffer is considered to be terminated when all the words or bytes in the assigned memory locations have been sent or received. Unbuffered operations are where data is exchanged **within** the computer between the CPU and various parts of the computer. Unbuffered operations do not establish limits when transferring information. Buffered operations, on the other hand, are for the expressed purpose of transferring information to and from the computer and an external device; they have established buffered limits. For example, addresses 00₈ through 17₈ in memory maybe set aside to receive data into the computer. A buffer can also be called a **frame**.

Sequencing

The I/O processor executes I/O commands using sequencing circuits in a manner similar to the CPU. Like the CPU, the I/O processor's sequencing circuits control the order in which events will be executed based upon the translated function code and modifying designators. To complete a particular I/O command, CPU instruction, or maintenance console/equivalent action (if available) may require the I/O processor to run one or more of the available sequences. A processor

may have up to six sequences depending on the design of the computer.

I/O Interface Circuits

The CPU interfaces with the I/O processor through the CPU's I/O instructions. These instructions cause the initiation of I/O operations. For computers with an IOC, the instructions allow the CPU to access the RTC or the monitor clock. This communication is done via the bus system. The communications lines include some of the following:

- Request lines (initiate I/O instruction)
- I/O(C) select lines
- Data lines
- Data ready
- Interrupt requests

I/O Memory Reference

The I/O processor references main memory during specific sequences such as an instruction or a maintenance console/equivalent action (if available). The bus allows this to be performed asynchronously. The I/O processor acquires I/O commands, output data, and operands from main memory and presents the information for storage into a main memory location over a bus. Some of the lines of communication include the following:

- I/O memory selection
- I/O read reference
- I/O write reference

I/O Control Memory

I/O processors can also use an I/O control memory, which is used primarily by mainframe and minicomputers containing an IOC. I/O control memory words are set aside in main memory to control data transfers for I/O buffer functions. I/O control memory is capable of handling parallel or serial information.

PARALLEL OPERATIONS.— In parallel operations, each I/O channel has its own block of memory addresses (usually 16). They include blocks for input, output, external function, and external interrupt operations. Some of the items included in parallel operations are as follows:

- Buffer control words (BCWs) —Buffer control words control the type and number of words or bytes that are to be transferred by the pending operation. Transfers include 8-bit bytes, 16-bit single words, and 32-bit words.

- Buffer address pointers (BAPs) —Buffer address pointers specify the next memory address, within the buffer, for a transfer to take place.

- Chain address pointers (CAPs) —There is one chain address pointer for each input and output chain of a channel. Each CAP specifies where in memory the IOC can find the next chaining instruction.

SERIAL OPERATIONS.— Serial operations are affected by character size (5 to 8 bits), parity selection (odd, even, or none), baud rate (50 to 9600 baud), and synchronous (sync) or asynchronous (async) interfacing. Some of the items included in serial operations are as follows:

- Monitor words —Monitor words are used to store characters for comparison with received (input) data characters.

- Suppress word —A suppress word contains a code that is used to remove specific characters from the serial transmission stream.

CONTROL MEMORY OPERATIONS.— The contents of control memory are accessed and modified through the use of IOC command or chaining instructions. The exception is actual data transfers in which the IOC logic updates control memory for each word or byte transferred. The basic operations that deal with control memory are the following:

- Initiate transfer (command or chain) —Initiate transfer loads the input or output BCW and BAP in control memory for the channel specified and initiates the input or output transfers.

- Load/write control memory (chain) — Load/write control memory is used to load or write data into single control memory word locations.

- Store control memory (command or chain) — Store control memory is used to write the contents of a specified control memory address into a memory address for CPU processing.

- Set/clear flag (chain) —A set/clear flag is used to set or clear (zero) specified bits or bit groups in control memory or main memory locations or the channel status word. It is also used to set or clear the test bit in the channel status word for conditional jumps.

- Search for sync/set suppress/set monitor (chain) —The search for sync/set suppress/set monitor enables or disables sync, monitor, and suppress capabilities indifferent serial configurations.

- Set/clear discrete (command and chain) — Set/clear discrete is similar to set/clear flags except that the set/clear discrete deals with serial interfaces exclusively. It is used to turn on or turn off specific serial channel signals such as data terminal ready.

- Channel control (command or chain) —Channel control performs a variety of single and multichannel functions. It can be used to master clear a single or all IOC channels, input or output. It is also used to enable or disable all, low priority, or a single channel's interrupts (external or class III interrupts).

CATEGORIES OF I/O OPERATIONS

There are two ways that the I/O section will handle the transfer of data between the computer and the external units: direct CPU/external device (**direct CPU interface**) communication and **direct memory access (DMA)**. Each method has its advantages and disadvantages. We begin with direct CPU/external device communication.

Direct CPU Interface

With direct communication, also called **accumulator based I/O**, the peripheral devices are tied directly into the CPU communication bus (control bus, data bus, and so forth). In a simple I/O scheme, the CPU handles all I/O transactions by executing one or more instructions for each word of information transferred. Three techniques are used: memory mapped I/O, polled I/O, and interrupt driven I/O.

MEMORY MAPPED I/O.— In memory mapped I/O, the CPU accesses the I/O device by placing appropriate addressing information on the bus. The addressing information uniquely identifies the device and possibly several addressable locations within the device. Thus an addressable location in an I/O device might be treated as a memory location in the computer. This enables the CPU to transfer data to and from the I/O device in the same way as main memory transfers. The following is an example:

Each I/O channel is assigned four memory addresses in main memory or in logic circuitry (registers) that replaces or overlays four sequential main memory addresses. These four addresses or registers are used to store the following data:

- address n — External Interrupt Code Word
- address n + 1 — Input Data Word
- address n + 2 — Output Data Word
- address n + 3 — Channel Control/Status Word

These addresses also allow the IOC/CPU to perform interrupt driven or polled I/O operations. Addresses n + 1 and n + 2 can be used as single word buffers for polled operations with the channel status word (n+ 3) acting as the status word for the CPU to periodically sample (poll).

POLLED I/O.— In polled I/O, the CPU must regularly check— or **poll** — each channel or port in turn to determine if it has information for input or is ready to accept data for output. A flag register can be used to check the port’s status. Polling is time consuming. The CPU must pause between executing processing instructions and poll of each port. A port’s status is examined in case action is required by the computer. We use a keyboard as an example of polled I/O. Figure 7-12 shows a read operation. The CPU reads or receives 8-bit encoded characters as they are typed on the keyboard. The CPU is programmed to read the input characters from an external device, in this case a keyboard. The keyboard inputs parallel 8-bit character codes for each depression of the keys. Characters are entered slowly as compared to the CPU’s ability to

process them. The dedicated CPU has to wait until the next character is entered each time.

The CPU is programmed with what is known as an **I/O wait loop**. As the CPU executes the loop instructions, it periodically (say 20 times a second) checks the status code from the keyboard to see if a character has been entered. A data register, INBUF, in the keyboard interface receives the character data from the keyboard. It holds the data until read by the CPU. A status register, INSTATCODE, indicates whether there is a new character in the INBUF register. By continuously testing the status register, the CPU detects when the code for a data entry is present. The CPU then executes the instructions to transfer the data from the data register to the specified location in the computer. Once this has completed, the CPU returns to the wait loop and polling process. The same procedure can be used for output or write operations. Figure 7-13 shows an output operation. In this case, the data is moved from a computer location to the data output buffer of the output device.

One of the disadvantages with polled I/O is that it involves the CPU throughout the input/output process. This is wasteful of CPU time. The CPU spends time executing input/output instructions that it could be spending performing other operations. Direct CPU interface has its place, particularly in small computers that are not concerned with high-speed operations and processing very large amounts of data. Most of the larger computers, however, use interrupt driven I/O.

INTERRUPT DRIVEN I/O.— The interrupt technique requires more complex hardware and software, but makes far more efficient use of the computer’s time and capacities. In an interrupt driven I/O, the I/O section itself is capable of accessing memory via the computer communication buses. The I/O processor can, while conducting I/O operations,

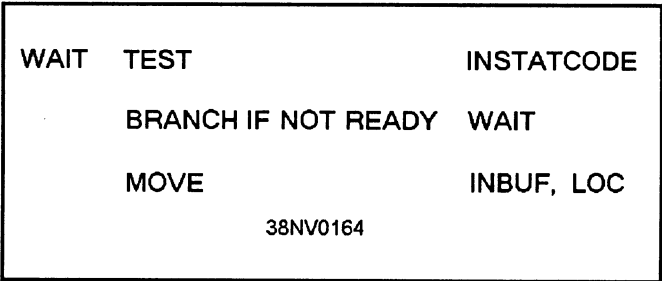


Figure 7-12.—Polled I/O; read operation,

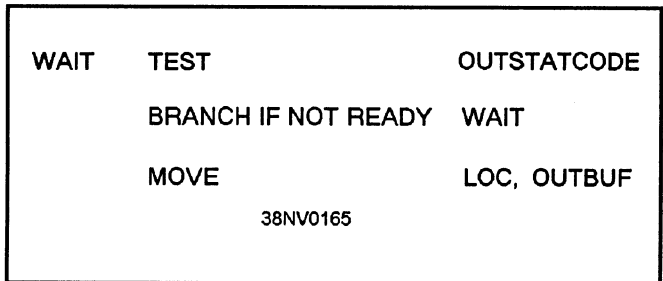



Figure 7-13.—Polled I/O; write operation.

read data from memory (**output**) or write data into memory (**input**). The CPU still provides overall control of the I/O operations, but it is not directly involved in the actual data transfers between memory and the external equipments. When the I/O section is capable of memory access, the CPU provides I/O commands to an **I/O controller (IOC)** or **processor** and then goes about processing other necessary operations. The IOC in turn controls all I/O operations and **interrupts** the CPU operation when necessary to inform it of event completion or problems with an I/O interface channel or external device.

With this method, the CPU concentrates on its essential business of processing information. We use the keyboard again as our example. The keyboard is ready to input characters. The keyboard interface signals the CPU when a valid character is available in its INBUF buffer. The CPU is performing some computational task, when the keyboard sends an interrupt request that generates an interrupt in the CPU. When the interrupt request arrives, the CPU leaves its current task, but not before making arrangements to save all the data from computations just previous to the interruption. The CPU leaves its current task and executes the appropriate service routine. In this case it receives the input from the keyboard interface and promptly sends it to the desired location in the computer. When the information has been routed to its desired location and the input operation has been completed, the CPU returns to its previous task. Review chapter 5 of this volume for a detailed discussion of the interrupt process.

An interrupt request can occur at any time. To avoid confusion, most computers use a priority system for requests in the event that two or more interrupts arrive simultaneously. Interrupt driven I/Os use a priority system to honor requests and interrupts. The priority system is divided into channel and function priorities. The channel priority performs priority determination of requests and interrupts based on the channel number. Figure 7-14 reflects channel priority of a computer with 16 channels. Notice how they are grouped and prioritized. Function priority determines the order of honoring requests and interrupts when channel priority honors more than one request per channel. See figure 7-15.

| GROUP | CHANNEL (OCTAL) | PRIORITY |
|-------|-----------------|-------------------------------------------------------------------------------------|
| 0 | 0 |  |
| | 1 | |
| | 2 | |
| | 3 | |
| 1 | 4 | |
| | 5 | |
| | 6 | |
| | 7 | |
| 2 | 10 | |
| | 11 | |
| | 12 | |
| | 13 | |
| 3 | 14 | |
| | 15 | |
| | 16 | |
| | 17 | |


38NV0166

Figure 7-14.—Channel priority determination.

Depending on the type of computer, interrupts are categorized and the program can be written to meet specific requirements when an interrupt occurs. Some interrupt requests cannot be ignored. For example, when a power failure interrupt occurs, the computer is given the needed time to save information before the computer system shuts down.

Direct Memory Access (DMA)

When the CPU is directly involved in each of the I/O data transfers, it slows down the process of moving information in and out of the computer. The use of

| FUNCTION | PRIORITY |
|--------------------|---------------------------------------------------------------------------------------|
| EXTERNAL INTERRUPT |  |
| EXTERNAL FUNCTION | |
| OUTPUT DATA | |
| INPUT DATA | |

38NV0167

Figure 7-15.—Function priority determination.

direct memory access (DMA) gives the computer an advantage-speed. It allows information to be moved quickly in and out of memory without the intervention of the CPU. DMA is given control and takes over from the CPU as director of electronic traffic on the computer's network of communication buses. It allows blocks of information to be transferred directly in and out of memory and from and to an external device without any CPU intervention. Information is transferred at a speed compatible with the speed of the external device. Therefore, the use of DMA would be advantageous when using a high-speed external device, such as a magnetic disk. The DMA acts the same as an I/O processor; it is just another method to control the flow of information.

A DMA controller is usually placed between the external device and the computer's bus. The controller uses circuits consistent with the computer's other major functional areas. The controller consists of several functional parts. Two counter registers are used. One generates the next main memory addresses from which information is read or in which it is stored. This counter register is incremented by successive information transfers. The second counter keeps track of the number of information words that are remaining to be transferred. A **data** register serves as a buffer between main memory and the external device. And of course, the **control** circuits, will control DMA operations. Other registers are provided for more complex external devices.

In its most usual form, a DMA assumes command of the computer's bus when the DMA controller receives an interrupt signal from an external device. It then gives the CPU a hold/suspend operations message. The CPU will respond with a hold-acknowledge signal. It turns over control of the bus and then, in effect, takes a short break. Meanwhile, the DMA controller moves information between main memory and the I/O external devices and independently carries out the I/O transfers. The DMA controller will inform the CPU when it is finished with an interrupt. During DMA operations, the CPU performs other tasks. If the CPU and the DMA controller try to access main memory simultaneously, the DMA has priority.

TOPIC 4—INPUT/OUTPUT INTERFACING

Input/output (I/O) interfacing is affected by many factors. Among them are the method of connection, serial or parallel interfacing, and the type of equipment the computer is interfacing with. Input/output operations allow the computer to communicate with an assortment of external devices. Most computers use an I/O processor of some sort, so we concentrate our discussion in that area. The external devices are connected to the I/O processor via **I/O channels** or **ports**. An I/O channel or port is nothing more than the wiring necessary to interconnect the computer's I/O processor with one or more external devices. The type of interfacing used will dictate the wiring of each channel or port. Computers may have a small number of channels or ports with multiple equipments connected to each channel, or they may, particularly in larger computers, have a number of I/O channels with limited numbers or types of external equipments on each channel or port.

METHODS OF CONNECTIONS

There is a great deal of variety not only in the types of external devices but also in the methods of connecting them to a computer. One thing that computer external devices have in common is that they communicate with the computer indiscrete binary data. The function of the external equipment may be to convert that data to other forms, but when a data exchange is done over I/O channels, the data exchange is in some form of binary data. We now look at two methods of connecting the external equipments where more than one external device is involved: daisy chaining and independent request control.

Daisy Chaining

When more than one peripheral device is connected to a single port/channel, a technique called daisy chaining is used. When daisy chained, the peripheral devices receive or transmit information over a common path. A separate set of addressing or control lines is used to identify (address) specific devices and to control the transmission or reception of information. When the CPU dictates the use of the computer's bus, there is no difficulty in deciding which external device will have access to the computer's bus.

But in more complex situations, such as DMA transfers, simultaneous requests for the computer's bus may be made by two or more external devices. Then a

preset method decides the order in which the devices can use the computer's bus. Refer to figure 7-16 as you read. An I/O controller of some type will correspond with the external devices. When an external device requests control of the bus, it signals the controller by activating the common **bus request** line. The devices on the line have **ORed** connections. The controller **acknowledges** the use of the bus on a separate line. The I/O controller will scan the chain with an acknowledge signal until it reaches the external device that requested the bus. The external device stop further propagates the acknowledge signal and accesses the bus. When two or more devices request control of the bus, the external devices closest to the I/O controller will be granted access to the bus first. Thus the order of connection on the daisy establishes the priority of which external devices are given access to the computer's bus.

Independent Request Control

Independent request control (fig. 7-17) offers a faster and more flexible way to the control bus requests. In this method, separate lines are used for the **request** and **acknowledge** lines. The I/O controller assigns **priority** to each external device, which can be fixed or programmable. A combination of the two methods produces greater flexibility when dealing with simultaneous requests, particularly when dealing with interrupt driven I/O. When signaled on a common interrupt request line, the CPU can poll all external devices in a **predetermined** order to find which external device needs to be serviced. This method is entirely software. Generally speaking, computers that use a request and acknowledge system, prioritize the **functions** and the **channels**. Some of the functions, in descending order, include the following:

- External Interrupt
- External Function
- Output Data
- Input Data

The channels/ports are also prioritized. Equipments are assigned a channel/port and usually the channel with the highest number will be serviced first by the computer. Figures 7-14 and 7-15 apply.

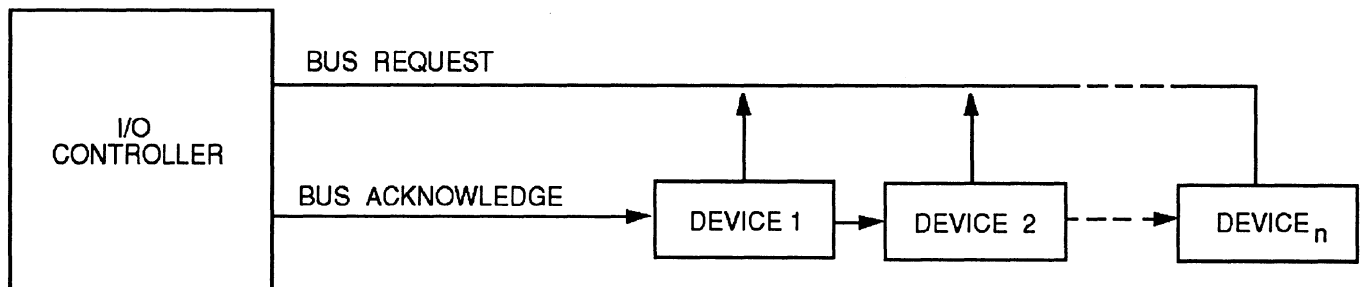
I/O INTERFACING STANDARDS

There are two major types of computer/external equipment communication formats: **serial** and **parallel**. The communication **formats** are governed by the **standard** that is identified by the **interface**. The interfacing standards provide valuable information. As a general rule the standards can be divided into four categories: mechanical, electrical, functional, and procedural. The standards can provide other standards that must be adhered to but do not fall into any one of these four categories.

- Mechanical —The mechanical portion takes into account such things as the type of connectors to be used, the number of pin connections in the connectors, and the maximum cable lengths allowed.

- Electrical —The electrical characteristics include the allowable line voltages and the representations for the various voltage levels.

- Functional —The functional interface specifies such things as which signals-timing, control, data, or ground leads—are to be carried by each pin in the connector.



35NVM030

Figure 7-16.—Connecting external devices in a daisy chain.

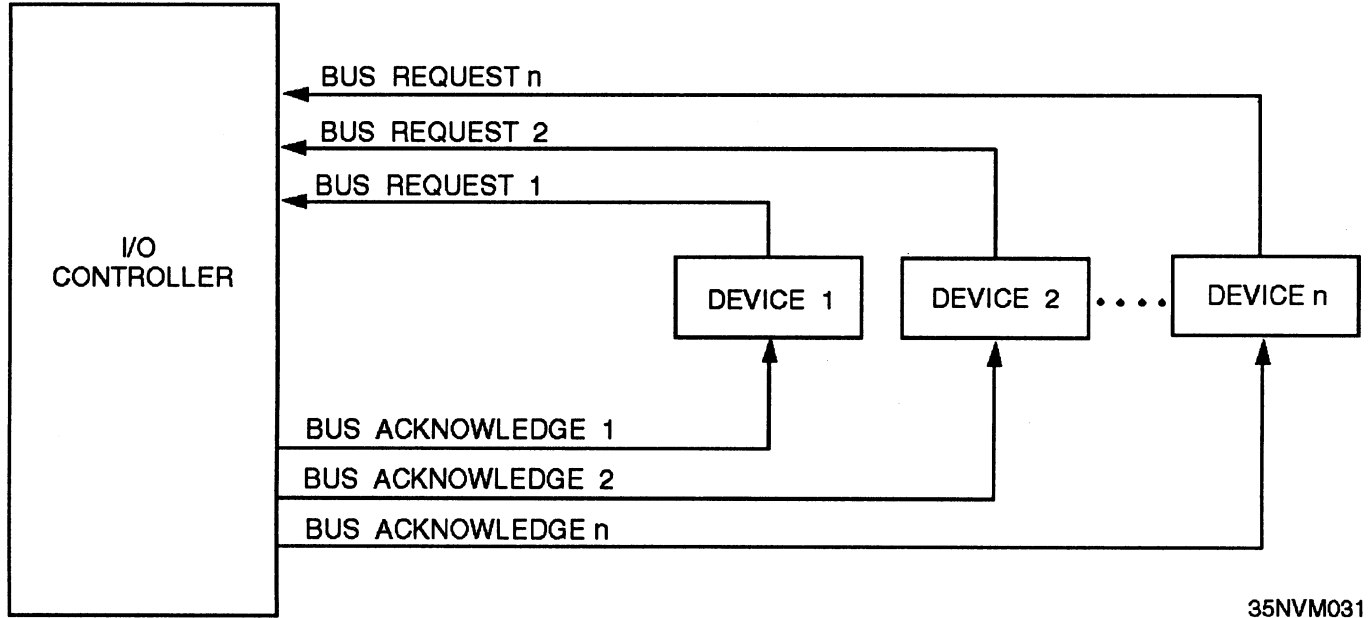


Figure 7-17.—Independent request control.

• **Procedural** —The procedural characteristics define how signals are to be exchanged and the environment necessary to input and output data.

No matter the format, **I/O interfacing components** are generally used by most computers regardless of the computer type.

I/O INTERFACING COMPONENTS

The computer's I/O processor, regardless of the type of computer and regardless of the type of format (serial or parallel) must ensure that the voltage levels between the computer and the external equipments are compatible. The primary circuitry that accomplishes this is located on an I/O pcb or modules/pcb's that make up an IOA. Some of the primary I/O interfacing hardware includes universal receiver transmitters, line drivers, and line receivers.

Universal Receiver-Transmitters

Within a digital computer, the data is transferred internally using a parallel format. All the bits of a byte

or memory word are exchanged simultaneously between registers, buses, and other computer logic. For the data to be communicated over a serial channel, it must be converted from parallel to a serial bit stream. Universal receiver-transmitters come in three types: universal asynchronous receiver-transmitters (UARTs), universal synchronous receiver-transmitters (USRTs), and universal synchronous/asynchronous receiver-transmitters (USARTs). A UART, USRT, or USART may be built into the computer or added as part of an I/O pcb or serial interface board. Modern UARTs, USRTs, or USARTs may consist of a single IC chip.

We take a look at a USART as an example of this type of logic assembly. The USART is designed to function as a peripheral device to the microprocessor. The microprocessor transmits byte-oriented data (data and command/control words) to the USART and receives byte-oriented data (data and status words) from the USART. The actual conversion from serial to parallel or parallel to serial is performed by the USART and is transparent to the microprocessor. The standard

USART chip (fig. 7-18) is composed of logic circuits, which are connected by an internal data bus. The logic circuits are read/write control logic, modem control, data bus buffer, transmit buffer, transmit control, receive buffer, and receive control.

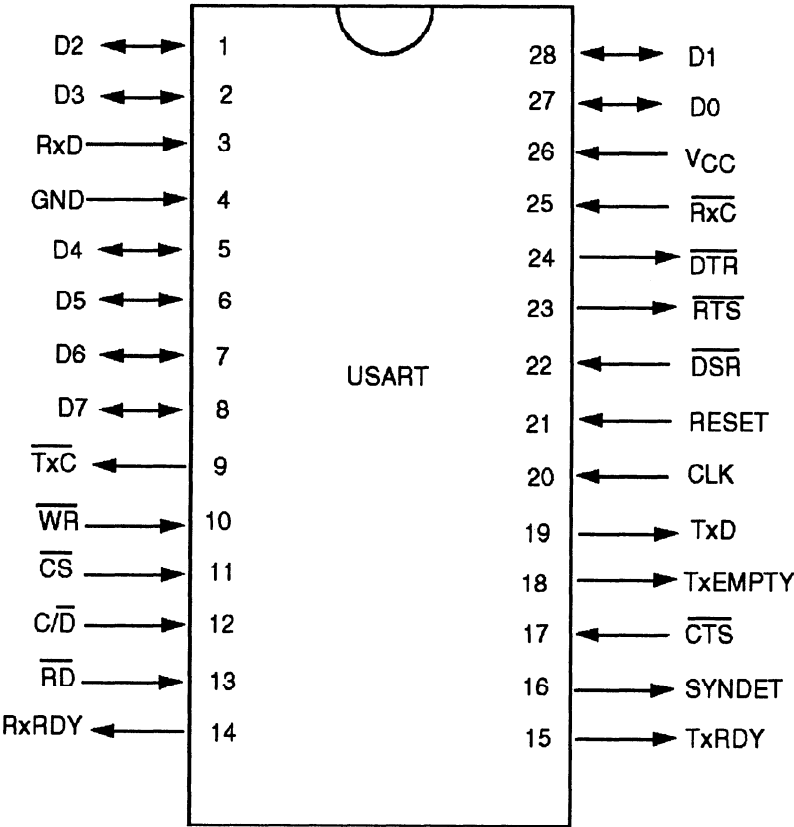
The CPU communicates with the USART over an 8-bit bidirectional tristate data bus. The USART is programmable, meaning the CPU can control its mode of operation using data bus control and command words. The read/write control logic then controls the operation of the USART as it performs specific asynchronous interfacing.

READ/WRITE CONTROL.— The read/write control logic accepts control signals from the control bus and command or control words from the data bus. The USART is set to an idle state by the RESET signal or control word. When the USART is IDLE, a new set of control words is required to program it for the applicable interface. The read/write control logic receives a clock signal (CLK) that is used to generate internal device timing.

Four control signals are used to govern the read/write operations of the data bus buffer. They are as follows:

- The CHIP SELECT (CS) signal, when true, enables the USART for reading/writing operations.
- The WRITE DATA (WD) signal, when true, indicates the microprocessor is placing data or control words on the data bus to the USART.
- The READ DATA (RD) signal, when true, indicates the microprocessor is ready to receive data or status words from the USART.
- The CONTROL/DATA (C/D) signal identifies the write operation transfer as data or control words, or the read operation transfer as data or status words.

MODEM CONTROL.— The modem control logic generates or receives four control or status signals used to simplify modem interfaces. They are as follows:



| | |
|---------|----------------------|
| D0-D7 | DATA BUS (8 BITS) |
| C/D̄ | CONTROL/DATA |
| RD̄ | READ |
| WR̄ | WRITE |
| CS̄ | CHIP SELECT |
| CLK | CLOCK PULSE (TTL) |
| RESET | RESET |
| Tx̄C | TRANSMITTER CLOCK |
| Tx̄D | TRANSMITTER DATA |
| Rx̄C | RECEIVER CLOCK |
| RxD | RECEIVER DATA |
| RxRDY | RECEIVER READY |
| TxRDY | TRANSMITTER READY |
| DSR̄ | DATA SET READY |
| DTR̄ | DATA TERMINAL READY |
| SYNDET | SYNC DETECT |
| RTS̄ | REQUEST TO SEND DATA |
| CTS̄ | CLEAR TO SEND DATA |
| TxEMPTY | TRANSMITTER EMPTY |
| VCC | +5 VOLT SUPPLY |
| GND | GROUND |

35NVM032

Figure 7-18.—Universal synchronous/asynchronous receiver transmitter (USART).

- **Data Set Ready (DSR)** —A data set ready is sent from the computer to the external device to notify the external device that the computer is ready to transmit data when HIGH.

- **Data Terminal Ready (DTR)** —A data terminal ready is sent from the external device to the computer to indicate that the external device is ready to receive data when HIGH.

- **Request to Send (RTS)** —A request to send is sent from the external device to the computer to indicate that the external device is ready (HIGH) or busy (Low).

- **Clear to Send (CTS)** —A clear to send is sent from the computer to the external device as a reply to the RTS signal.

TRANSMIT BUFFER/TRANSMIT CONTROL.— The transmit control logic converts the data bytes stored in the transmit buffer into an asynchronous bit stream. The transmit control logic inserts the applicable start/stop and parity bits into the stream to provide the programmed protocol. A start bit is used to alert the output device, a printer for instance, to get ready for the actual character (bit). The signal is sent just prior to the beginning of the actual character coming down the line. A stop bit is sent to indicate the end of transmission. The parity bit is used as a means to detect errors; odd or even parity maybe used.

RECEIVE BUFFER/RECEIVE CONTROL.— The receive control logic accepts the input bit stream and strips the protocol signals from the data bits. The data bits are converted into parallel bytes and stored in the receive buffer until transmitted to the microprocessor.

Line Drivers/Receivers

We discussed line drivers/receivers in chapter 4. Their basic function is to drive and receive (detect) the digital signal sent or received over a cable to other external equipments (including computers). The line drivers/receivers are designed to send and receive signals over short and long distances using serial or parallel **format**. Large voltages or currents are generated from small voltage or current using TTL or MOS circuitry. The two types most commonly used include **single-ended** and **differential**. The voltage levels and current amounts sent and received are dictated by the **interface**. The voltage and current characteristics required are also dictated by the **interface**. We discuss the voltage levels and some of

the characteristics when we cover I/O channel/port configurations that include the various interfaces.

I/O INTERFACE FORMATS

There is a variety of serial and parallel I/O channel formats that you may encounter as a technician. Do not take for granted the type of interface a computer uses. A single different pin in a connector or a different voltage level used by a computer can make a vast difference when you are performing maintenance. Your computer's technical manual will provide the standards to be used with the cabinet and cable connectors. They will match the standards that govern the requirements for parallel and serial interfacing. Table 7-1, from MIL-STD-2036, *General Requirements For Electronic Equipment Specifications*, provides you with some of the accepted standard external interfaces. We do not cover the General-Purpose Interface Bus (GPIB), Fiber Distributed Data Interface (FDDI), and TACTICAL. Other interfaces used but not listed in the table include RS-449, Centronics Parallel, ST-506/412, Enhanced Small Device Interface (ESDI), Integrated Drive Electronic (IDE), and Enhanced Integrated Drive Electronics (EIDE). We discuss signal designations in more detail later in this topic under serial and parallel I/O operations. First, let's look at the various interfaces and some of their applications and any unique characteristics. As stated, each interface is governed by a standard.

Table 7-1.—Standard External Interfaces from MIL-STD-2036

| Table I. Standard External Interfaces | | |
|---------------------------------------|-----------------|------------------------------|
| INTERFACE | STANDARD | FORMAT |
| NTDS INPUT/OUTPUT | MIL-STD-1397 | Digital (Parallel/Serial) |
| SCSI | ANSI X3.131 | Digital (Parallel) |
| RS-232 | EIA RS-232 | Digital (Serial) |
| RS-422 | EIA RS-422 | Digital (Serial) |
| GPIB | IEEE Std 488.1 | Digital (Parallel) |
| TOKEN RING | IEEE 802.5 | Digital (Serial) |
| ETHERNET | IEEE 802.3 | Digital (Serial) |
| FDDI | ANSI X3T9.5 | Digital fiber optic (Serial) |
| TACTICAL | MIL-STD-188-200 | Analog |

NTDS Input/Output (MIL-STD-1397)

The NTDS input/output interface is probably one of the most versatile of formats because it is designed to handle either parallel or serial formatted information, depending on the type of computer and its I/O requirements. This interface specifies three I/O control and data signal categories. We cover the first two under parallel and serial operations later in this topic. The categories include:

- Category I —Computer to external device
- Category II —Computer to computer, intercomputer (IC)
- Category III —External device to external device

Within this standard, there are nine types of formats (A through H and J). They include both serial and parallel formats as described in the following paragraphs.

TYPE A (NTDS) SLOW.— Type A transfers parallel data of up to 41,667 words per second on one cable. This type interface uses 0 vdc (logical 1) and -15 vdc (logical 0) to transmit bit groupings of 16, 30, or 32 bits, depending on the type of computer. The relatively large voltage change between logic states, with its inherent time delays, limits the speed of data transmission. Type A can transmit digital signals up to 1000 feet. It is most frequently used in large mainframe and some minicomputers to interface with equipment found in the data processing, display, and communication subsystems. Type A uses a request and acknowledge protocol process. It transfers control and data words using two cables: one input and one output for the same channel. You may, however, encounter a few devices that use input only or output only portions of an NTDS slow channel. Type A signal designations for input and output include the following:

- EIE —External interrupt enable
- IDR —Input data request
- EIR —External interrupt request
- IDA —Input data acknowledge
- EFR —External function request
- EFA —External function acknowledge
- ODR —Output data request
- ODA —Output data acknowledge

TYPE B (NTDS) FAST.— Type B transfers parallel data of up to 250,000 words per second on one cable. This type interface uses 0 vdc (logical 1) and -3 vdc (logical 0) to transmit bit groupings of 16, 30, or 32 bits depending on the type of computer. Type B can transmit digital signals up to 300 feet depending on the type of cable used. It is most frequently used in large mainframe or some minicomputers to interface with equipment found in the data processing, display, and communication subsystems. Type B uses a request and acknowledge protocol process. It transfers control and data words using two cables: one input and one output for the same channel. You may, however, encounter a few devices that use input only or output only portions of an NTDS fast channel. Type B uses the same input and output signal designations as type A.

TYPE C (ANEW).— Type C transfers parallel data of up to 250,000 words per second on one cable. This type of interface uses 0 vdc (logical 1) and +3.5 vdc (logical 0) to transmit bit groupings of 16, 30, or 32 bits, depending on the type of computer. Type C can transmit digital signals up to 300 feet depending on the type of cable used. It is most frequently used in large mainframe or some minicomputers to interface with equipment found in the data processing, display, and communication subsystems. Type C uses a request and acknowledge protocol process. It transfers control and data words using two cables: one input and one output for the same channel. You may, however, encounter a few devices that use input only or output only portions of an NTDS ANEW channel. Type C uses the same input and output signal designations as type A.

TYPE D (NTDS SERIAL)— Type D asynchronously transfers serial data using a 10 megabits per second (Mb/s) clock rate over a single coaxial cable. Two cables are required for bidirectional communications, a source line (computer to peripheral) and a sink line (peripheral to computer). The source line is used to transmit data and external functions, while the sink line is used to transmit input data and external interrupt codes. Type D transfers are accomplished using two types of bipolar pulse trains: (1) control frames and (2) control and data words. The actual input or output data is transmitted in 32-bit **information frames**. Control frames are three bits in length, a sync bit followed by two control bits. The signals required for input transfer will occur on the input channel (input request, input enable, and not ready) and the signals required for output transfer will occur on the output channel (output request, output enable, and not ready). A binary 1 will be a pulse of phase zero degrees and will be a high polarity followed by a low polarity.

A binary 0 will be a pulse of phase 180 degrees and will be a low polarity followed by a high polarity. Type D can transmit digital signals up to 1,000 feet.

TYPE E (NATO SERIAL).— Type E asynchronously transfers serial data of up to 10 million bits per second on single triaxial cable. Channel control is similar to NTDS parallel channels. This type interface uses a bipolar plus or minus 0.6 volt nominal (0.8 volt maximum). Type E can transmit digital signals up to 1,000 feet depending on the type of cable used. It is most frequently used in large mainframes to interface with external equipment found in the data processing subsystems (includes intercomputer communication). Interfacing with an external device uses a normal serial I/O interfacing: enable and request. The channel interface uses a SIS/SOS protocol, transferring control and data words using the following word transfers: external function, output data, external interrupt, and input data. The data (command or data) words are transmitted in serial bursts of up to thirty two 32-bit words (1,024 bits). The burst transmissions are coordinated using Sink Status (SIS) frames or Source Status (SOS) frames. The SIS frame is sent from the receiving device when it is ready to receive a burst. The SOS frame is sent by the transmitting device to coordinate and synchronize the burst transmission.

TYPE F (AIRCRAFT INTERNAL TIME DIVISION MULTIPLEX (TDM) BUS).— Type F transfers serial data up to one million bits per second over a distance of 300 feet. A logical 1 will be transmitted as a bipolar coded signal 1/0 (a positive pulse followed by a negative pulse). A logic zero will be a bipolar coded signal 0/1 (a negative pulse followed by a positive pulse). This type interface transmits bit groupings of 20 bits: data, sync wave form, and parity bit. It is most frequently used in large mainframes to interface with equipment found in the data processing subsystems. Type F uses a command/response protocol. Transfers include command, data, and status words over a single channel. This interface can handle up to 32 external devices on one channel; one device must be a bus controller.

TYPE G (RS-449).— Type G equates with the functional and procedural portions of RS-232. However, the electrical and mechanical specifications are covered by RS-422. Type G is intended to transfer serial data above 20 kilo bits per second and up to 2 million bits per second over a single cable. Type G can transmit data up to 200 feet. Signals are divided between 37-pin and 9-pin connectors, and the ground and common signals are handled separately for each

cable. Type G can send asynchronous serial data up to 9600 bits per second. This type of interface is used to transmit bit groupings of 8, 16, or 32 bits depending on the type of computer. Type G can be used in mainframe and microcomputers. Type G uses primarily a command and response protocol.

TYPE H (HIGH-SPEED PARALLEL).— Type H transfers parallel data of up to 500,000 words per second on one cable. This type interface uses 0 vdc (logical 1) and +3.5 vdc (logical 0) to transmit bit groupings of 16, 30, or 32 bits depending on the type of computer. Type H can transmit digital signals up to 300 feet. It is most frequently used in large mainframes to interface with equipment found in the data processing, display, and communication subsystems. Type H uses a request and acknowledge protocol process. It transfers control and data words using two cables—one input and one output for the same channel. It can also interface with external equipment having a type C interface. You may, however, encounter a few devices that use input only or output only portions of an NTDS slow channel. Type H uses the same input and output signal designations as type A.

TYPE J (FIBER OPTIC NATO SERIAL).— Type J is used for the fiberoptic implementation of type E. A type J fiber optic channel converts a type E serial bit stream into light pulses that are carried by a fiber optic cable to a receiving device that converts the light pulses back into a digital bit stream. For further details on fiber optics, refer to NEETS 24, *Introduction to Fiber Optics*.

Small Computer System Interface (ANSI X3.131)

The small computer system interface (SCSI) uses a digital parallel format. SCSI is pronounced “skuzzy.” The SCSI is an 8-bit parallel, high-level interface. High-level means that instead of a host computer asking for data by specifying a track, cylinder, and sector number, all it asks for is a logical sector number. The SCSI then translates the logical sector number into the actual disk location.

The SCSI also has other improvements over previous disk drive interfaces. For example, it can transfer data at rates up to 20 megabits per second, handle hard disk drives of almost any size, disconnect itself from the host computer’s bus while it processes requests, and daisy-chain up to eight units off of one controller.

The SCSI interface uses one 50-pin ribbon cable to connect the hard disk drive(s) to the controller card mounted on the host computer. Some computer manufacturers include the SCSI electronics in their motherboards and do away with a separate controller card.

RS-232 (EIA RS-232 and MIL-STD 188)

An RS-232 interface uses a serial format. It can be used for asynchronous and synchronous serial transfers. It can be used with mainframes, minicomputers, and microcomputers for communication with external equipments, particularly with microcomputer systems. RS-232 channels/ports are capable of transmitting from 50 to 19,200 baud of 7- or 8-bit asynchronous characters and 7- or 8-bit synchronous characters to 9600 baud. RS-232 limits cable transfers to 50 feet with a maximum transmission speed of 20,000 bits per second. In microcomputers and their external equipments, the Configuration of the channel/port is normally hardware controlled through the use of DIP switches. The number of bits per character (7 or 8), baud rate (110, 300, 600, 1200, 4800, 9600, or 19200), parity setting (odd, even, or no parity), and protocol selection (ready/busy or X-ON/X-OFF) are examples of controlled configuration parameters. Some computer systems allow for software control of these parameters but most peripherals that accept the RS-232 have a DIP switch configuration to make them compatible with a variety of computer interfaces.

RS-232 serial channel/port uses a 25-pin cable connector (DB-25) and transmits signal levels of +5 to +25 volts (HIGH or SPACE) and -5 volts to -25 volts (LOW or MARK). An RS-232 receives and recognizes transition difference of 6 volts (+3 volts and -3 volts) (fig. 7-19). A positive difference and more than +3

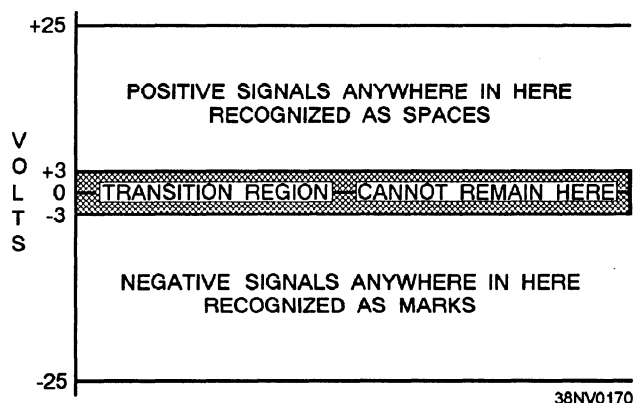


Figure 7-19.—RS-232.

volts indicates a HIGH and a negative difference and more than -3 volts indicates a LOW. Signal designations are discussed in serial I/O operations. An interface that uses RS-232 interface signals is VACALES (Variable Character Length Synchronous). It is synchronous to 32,000 baud transferring 1 to 16 bits.

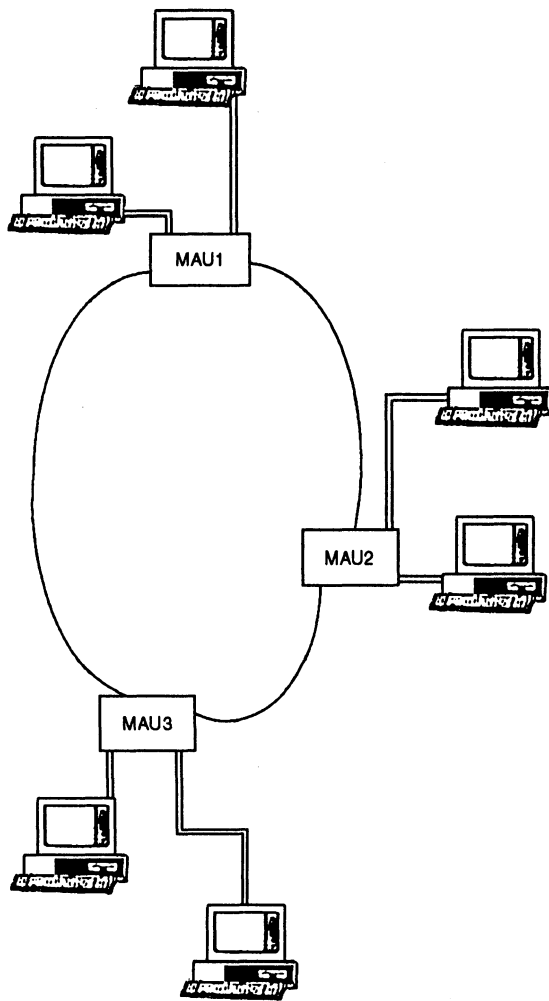
RS-422 (EIA RS-422)

The RS-422 interface uses a serial format. RS-422 uses RS-232 functional specifications. RS-422 uses two separate wires to allow transmission at a higher rate. This technique, called balanced circuitry, doubles the number of wires in the cable, but permits very high data rates and minimizes the problem of varying ground potential. The high data rates include up to 10 megabits per second in distances of meters and 100 kilobits per second at 1.2 kilometers. RS-422 grounding requirements are much less critical than RS-232. With the elimination of the grounding problem, the receiver transition period is narrower: .4 volt (+.2 volt and -.2 volt).

Token Ring (IEEE 802.5)

Token ring is used for work group solutions and work station intensive networks. It transfers serial I/O data. It has the ability to operate at a 4- or 16-megabits per second rate of data communication. It allows PCs and mainframes to operate as peers in the same network. In a token-passing ring network, a stream of data called a **token** circulates through the network stations when they are idle. A station with a message to transmit waits until it receives a **free** token. It then changes the free token to a **busy** token, and transmits a block of data called a **frame** immediately following the busy token. The frame contains all or part of the message the station has to send.

The system does not operate by having one station accept a token, read it, and then pass it on. Instead, the stream of bits that make up a token or message might pass through as many as three stations. Once a station becomes a busy station, there is no free token on the line. That means other stations must wait until the receiving station copies the data and the frame continues around the ring until it completes a round-trip back to the transmitting station. This guarantees that only one station at a time transmits data. A typical token ring (fig. 7-20) provides for unlimited expandability by use of multistation access units (MAUs) and hubs (concentrators).



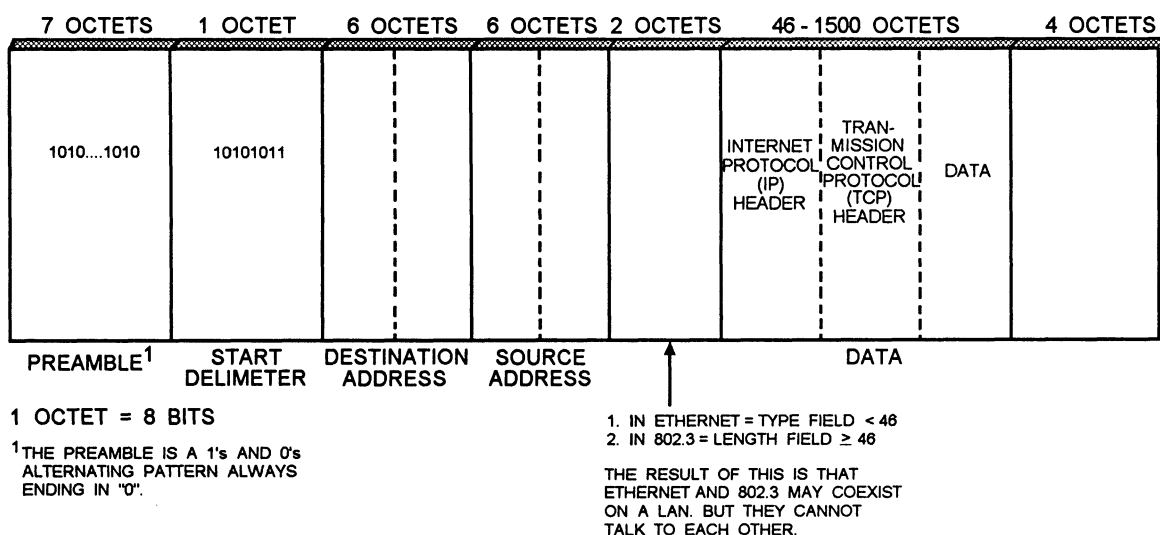
35NVM034

Figure 7-20.—A typical token ring network.

The recommended cable for a typical token ring setup is two pairs of twisted wire covered by a foil shield. Maximum cable length between the token-ring hub and the attachment point for the network node cannot exceed 150 feet. Provisions are also available for linking hubs through fiber optic cable. Connectors include “D” shell for the twisted pair wire and fiberoptic connectors (MIL-C-28876). Cabling for the token-ring prevents one bad cable from bringing down the entire system.

IEEE 802.3 (Ethernet DIX)

IEEE 802.3 is a specification that describes a method for computers and data systems to connect and share cabling (i.e., PC’s and mainframes). It transfers serial I/O data in a specific packet format (fig. 7-21). The IEEE 802.3 standard is commonly referred to as Ethernet. Although Ethernet and 802.3 share the same cable access mode (carrier sense multiple access), they differ in both physical implementation and actual packet make-up. Ethernet preceded IEEE 802.3 by almost 10 years. Ethernet was developed by Robert Metcalfe at Xerox’s Palo Alto Research Center. Ethernet is the forerunner of IEEE 802.3. Because of the differences in packet formation and physical construction of the equipment associated with each of these standards, the networking community currently follows the original Ethernet standard implementation by the DIX suffix (DIX stands for DEC, Intel, and Xerox, the original collaborators on the Ethernet standard).



ETFC0083

Figure 7-21.—802.3 and Ethernet packet formats.

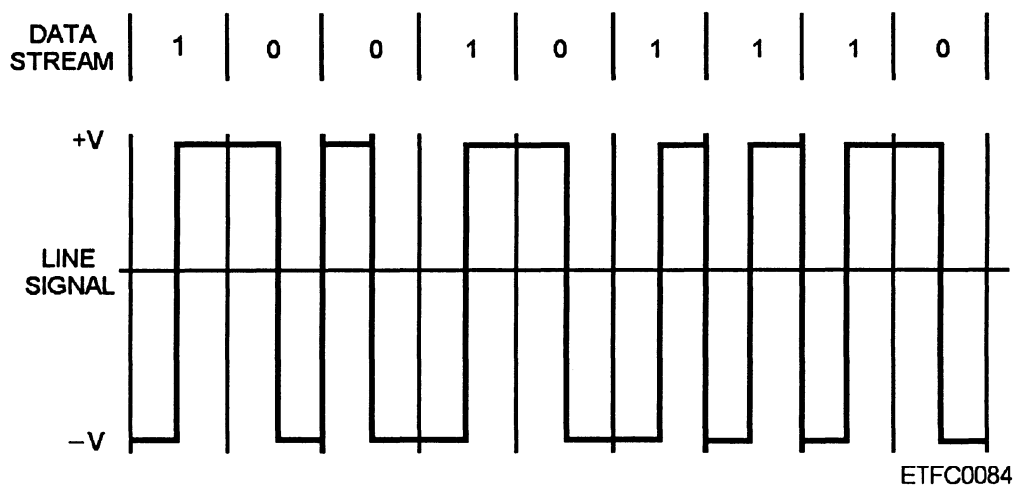


Figure 7-22.—Manchester code used in Ethernet.

Both Ethernet (DIX) and IEEE 802.3 can be used on the same data communications network, but they cannot talk to each other. Data in an 802.3 network is encoded using a Manchester code as shown in figure 7-22. The differences between an 802.3 packet and an Ethernet packet can be seen in figure 7-21. When viewing figure 7-21, pay close attention to the items directly below the vertical arrow in order to determine an Ethernet (type field <46) or an 802.3 (length field ≥46).

Continuous transitions of the Manchester code allow the channel to be monitored easily for activity. This is part of the Collision Detection/Collision Avoidance characteristics of Ethernet (DIX) and IEEE 802.3. This ability to detect activity allows stations to release the channel after using it for a short period of time, thereby increasing data transmission through-put.

Ethernet (DIX) and IEEE 802.3 may use a shielded coaxial cable (RG-58 A/U) to transfer serial data using baseband transmission at 10 megabits per second.

Baseband information implies data transmitted without the use of a carrier and with only one channel defined in the system. When a station is transmitting, it uses the entire 10 megabits per second. The data is transferred PC to PC using a daisy chain configuration (fig. 7-23). Thin Ethernet is used in smaller systems using an overall coaxial cable length of 600 to 1000 feet. Thin Ethernet 802.3 uses T-connectors (UG-274) to connect the PCs. Thick-net (RG-11, BIG YELLOW CABLE) is used in larger systems with overall shielded coaxial cable lengths of 500 meters. Thick-net networks employ a file server and a transceiver (fig. 7-24) connected together using 15-pin “D” shell connectors. Terminating resistors are used at the end of each T-connector to ensure proper operation. Ethernet (DIX) and IEEE 802.3 networks are also commonly implemented using shielded and unshielded twisted pair cable. Coaxial cable implementations are known as 10Base5 (RG-58) and 10Base2 (RG-11 Thick-net). Shielded and unshielded twisted pair cable networks are known as 10BaseT.

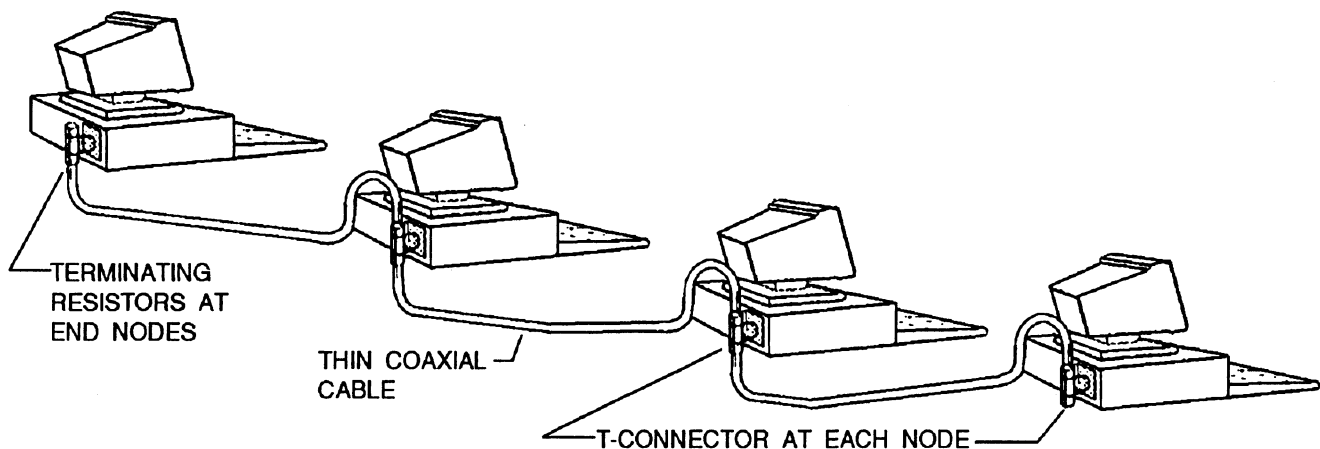
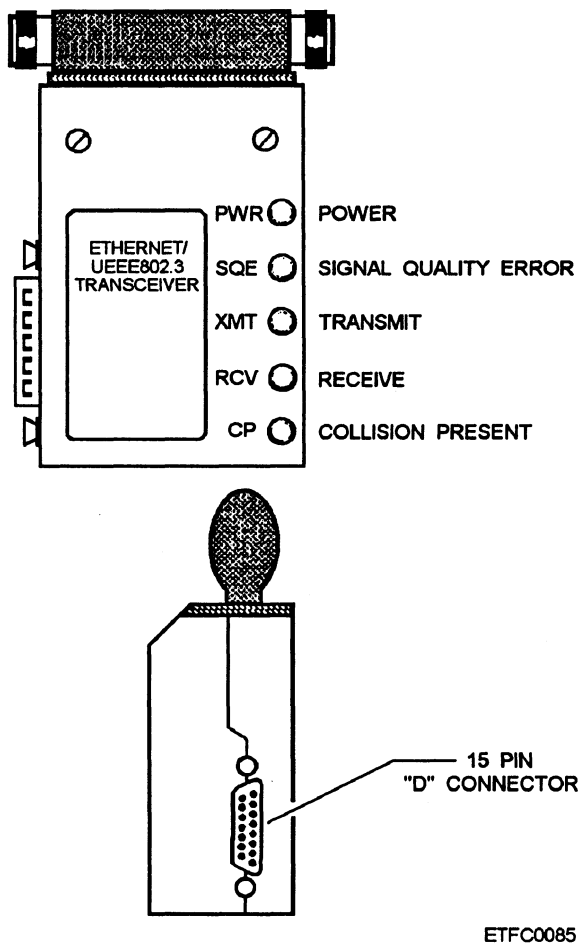


Figure 7-23.—Daisy chain in an Ethernet.

35NVM035



ETFC0085

Figure 7-24.—Ethernet/IEEE 802.3 transceiver.

Centronics Parallel

The Centronics compatible parallel channel is the alternate interface to the RS-232 on many microcomputer systems. This channel type is designed to transmit parallel 8-bit bytes over eight data lines simultaneously. The Centronics compatible channel is a single direction channel (output only) as far as data is concerned. Centronics Compatible Parallel uses a command/acknowledge protocol. There are several control signals sent to the receiving device and status signals returned from the receiving device. We cover signal designation under parallel operations of single cables.

ST-506/412

The ST-506/412 interface was developed by Seagate Technology, Inc. It is often used in the hard disk drives installed in older IBM-compatible desktop computers that have a maximum capacity of 125 megabytes. It is also the interface used to control most floppy drives today.

This is one of the interfaces where most of the electronics is actually on a controller card mounted in the host computer. With this interface, the controller card does most of the work (moving the magnetic head, spinning the disk, and so on). The controller card also cleans any data coming from the disk drive by stripping off the formatting and control signals that were used to store the data onto the hard drive.

A hard disk drive is connected to the controller card in the host computer via two ribbon cables (a 34-pin control cable and a 20-pin data cable). Floppy drives use only the 34-pin control cable to transfer both data and control signals.

When this interface was originally developed in 1981, its 5-megabits per second transfer rate was considered too fast. It was actually slowed down by a 6:1 interleave factor so it could operate with the computers being built at the time. With today's transfer rates pushing the envelop at 24 megabits per second, you can see that it is now one of the slowest interfaces.

Enhanced Small Device Interface (ESDI)

The enhanced small device interface (ESDI) is an optimized version of the ST-506/412 interface. The main difference is that with ESDI, most of the disk drive's interface electronics is located in the disk drive itself, rather than on a controller card in the host computer. The result is a much faster transfer rate and more hard disk capacity. ESDIs have a transfer rate of up to 24 megabits per second. And, they can handle disk drives with a maximum capacity of 1.2 GB (gigabyte).

The ESDI uses the same interface cables as the ST-506/412 interface, but that is where the similarity ends. With ESDI drives, only the clean data is sent to the controller card in the host computer. All formatting and control signals are stripped off at the hard disk drive.

Integrated Drive Electronics (IDE)

The integrated drive electronics (IDE) interface was developed as a result of trying to find a less expensive way to build computer systems. It includes all of the controller card electronics in the hard drive itself; thus, the hard drive does all the work.

The hard disk drive connects to the host computer's bus with a 40-pin ribbon cable. The ribbon cable connects directly to either a 40-pin connector on the host computer's motherboard or a 40-pin connector on

a small interface card that plugs into the host computer's motherboard. This interface offers transfer rates of up to 1 MB and can handle hard drives with a maximum capacity of 300 MB.

Enhanced Integrated Drive Electronics (EIDE)

The Enhanced Integrated Drive Electronics (EIDE) was developed from the IDE standard. New features available with EIDE include Plug-n-Play compatibility, increased maximum drive capacity, faster data transfers, and the ability to use a CD-ROM or tape drive with an the interface.

The IDE interface can address a hard drive with a maximum of 504MB. EIDE increases the maximum size of a hard drive by using an enhanced BIOS. The enhanced BIOS uses a different geometry when communicating with a program than it does when communicating with the hard drive. For example, the BIOS will tell a program that a hard drive with 2,000 cylinders and 16 heads is a drive with 1,000 cylinders with 32 heads. The BIOS controls the address translation to keep track of where the data is physically located on the hard drive.

The EIDE interface uses a Programmed Input/output (PIO) mode to transfer data from the drive. There are five PIO modes that can be set to control data transfers. PIO Mode 0 is the slowest with a cycle time of 600 nanoseconds. Pio Mode 4 has a cycle time of 120 nanoseconds, which is 16.6 megabytes per second. Most high-end hard drives will support Mode 3 or Mode 4 operations. Using the enhanced BIOS, the hard disk responds to the Identify Drive command with information concerning the PIO and DMA modes the drive can support. The BIOS will automatically set the PIO mode to match the capability of the drive. If a drive is set to a higher mode than it is capable of supporting, data corruption will occur.

I/O SERIAL DATA OPERATIONS

Serial data operations exchange information via a single path, line, or wire. The channel/port itself is made up of several wires, but only one is used to transfer the binary data. Bidirectional channels may use two wires for data, one for each direction or a single tristate bidirectional line. The remaining wires are used for device addressing and to provide the **protocol** (channel control) for information exchange. The data is in the form of an asynchronous or synchronous bit stream. The bit stream is made up of a sequential series of data and/or control pulses in one of these two mutually

exclusive formats. Serial data operations can use a minimum of 4 conductors and up to 37 conductors to perform serial data operations. Serial operations generally exchange information between **data communications equipment (DCE)** and **data terminal equipment (DTE)**. The DCE configured device is considered the controller for the interface. The DTE is either the computer or a channel controller. There are variations in the channel pin connections that depend on the device mode of operation (DCE or DTE).

Asynchronous Data Exchanges

Asynchronous data is also known as **character framed data**; only one character at a time is sent. Each character is composed of either 7 or 8 bits (depending upon the coding scheme used), and is identified by a start and stop bit. At the minimum, each character is preceded by a start bit and followed by one stop bit. Asynchronous data transmission protocol allows for a maximum of the following in sequence: one start bit, eight data bits, a parity bit, and one stop bit for each character to be exchanged.

The purpose of a start bit is to notify the modem that a character is being sent (or received). The bits that make up the character immediately follow the start bit. After all these bits have been transmitted, a stop bit is inserted to indicate the end of the character. Start and stop bits can immediately follow one another or there can be a period of idle time following the stop bit, depending upon the hardware device in use. During an idle condition, in which no characters are sent, a continuous MARK signal (equivalent to a logic 1) is transmitted for one bit time. Asynchronous transmission is normally used when transmission rates are between 600 to 2000 bps. The particular format used varies between computers and may be hardware or software controlled depending on the type of interface logic and devices used.

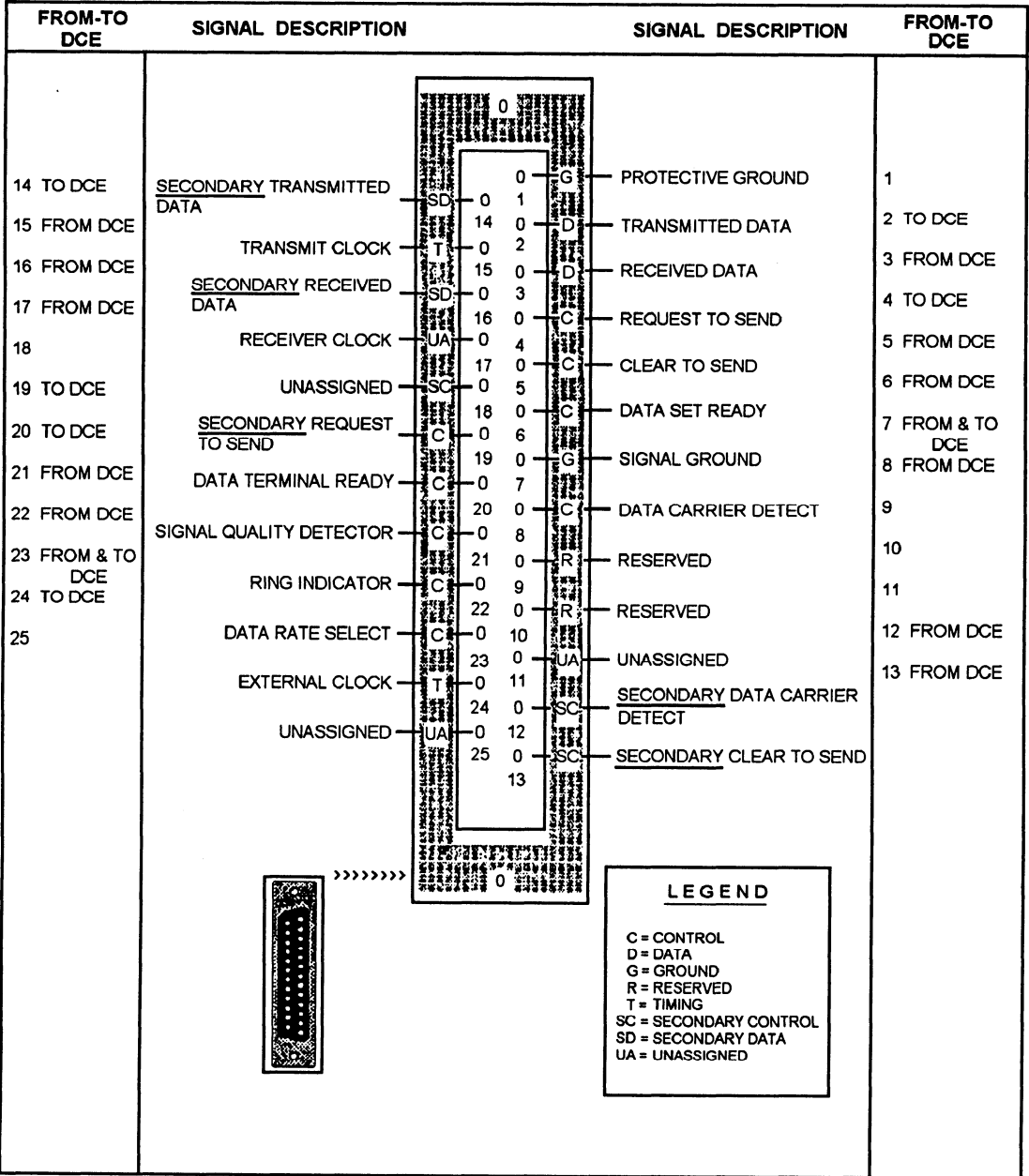
Synchronous Data Exchanges

When more speed is required for sending information, synchronous data exchanges fulfill the requirement. Synchronous data is also known as **message framed data**. The bit stream is divided into blocks of sequential bits grouped into individual messages, without the need for start and stop bits. Again, each character is composed of either 7 or 8 bits. There are two methods for controlling the exchange of messages. External control and timing signals may be

transmitted over the I/O channel control lines and used to synchronize the message transfer, or the message itself may be preceded and succeeded by a string of special synchronization (sync) characters. The sync characters allow the receiving device to frame and receive the message data. Messages preceded by sync pulses are followed by one or more special synchronization (sync) characters to indicate the end of a particular bit stream. Often several different types of messages are sent over the same channel. The message contents identify the type of message and the destination (addressed peripheral).

DCE/DTE Serial I/O Cable Signals

With serial operations, one cable will suffice to perform serial I/O operations with an external device. Each of the signal leads is assigned a specific function. These functions can be assigned one of four specific groupings: data (both primary and secondary), **control** (again, both primary and secondary), **timing**, and **ground**. Each of these groupings is indicated by a letter in figure 7-25 and is further described in the legend. We use an RS-232 as our example in this discussion. Although the connector itself is not specified in the standard, a 25-pin connector (such as the one shown in figure 7-25) has become the generally accepted



ETFC0086

Figure 7-25—A typical RS-232 female connector.

standard for implementing an RS-232 connection. Now, let's take a pin-by-pin tour of the RS-232 interface and look at the signals to see how they function.

RS-232 Pin Description

The 25 pins of the RS-232 have the following functions:

Pin 1, Protective Ground— is connected to the equipment's chassis and is intended to connect one end of a shielded cable, if such a cable is used. The shield of a shielded cable must **NEVER** be connected at both ends. Shielded cable is used to reduce interference in high-noise environments.

Pin 7, Signal Ground— is the common reference for all signals, including data, timing, and control signals. In order for DCE and DTE to work properly across the serial interface, pin 7 must be connected at both ends. Without it, the interface would not work because none of the signal circuits would be completed.

Pin 2, Transmitted Data

Pin 3, Received Data— Pins 2 and 3 are the pins of most importance; for if it weren't for the data that passes through them, the remaining pins would not be needed. Data is normally transmitted in the following manner. The DTE **transmits** data on pin 2 and **receives** data from the DCE on pin 3 as described in figure 7-25 and shown in figure 7-26. Figure 7-26 illustrates the absolute minimum wiring required under the RS-232 interface for normal DTE-DCE communication.

Pin 4, Request to Send

Pin 5, Clear to Send— Pins 4,5,6, and 20 are the handshaking signals. These pins establish the communications link. Normally terminals cannot transmit data until a clear to send transmission is received from the DCE.

Pin 6, Data Set Ready

Pin 20, Data Terminal Ready— Data set ready is used to indicate that the modem is powered on&d is not in a test mode (modem ready). In dial-data or dial up applications, data terminal ready is used to create the equivalent of an off-the-hook condition. When the modem is in an auto-answer mode, the DTR is activated in response to the ring indicator and tells the modem to answer the incoming call.

Pin 8, Data Carrier Detect— The modem activates the data carrier detect whenever it receives a signal on the telephone line of sufficient strength for reliable communications. Many types of DTE

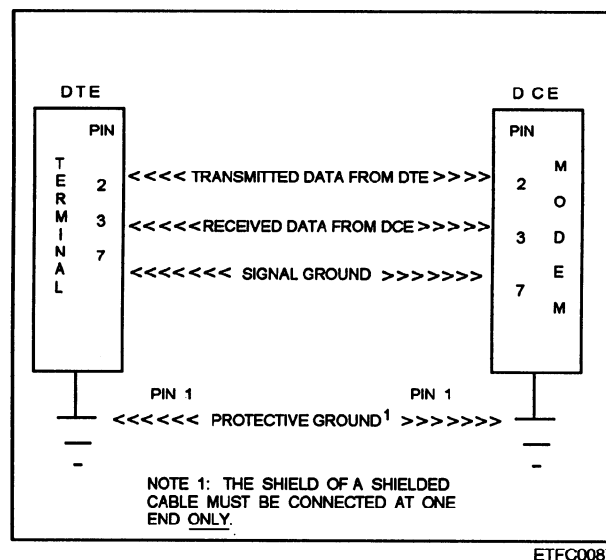


Figure 7-26.—A typical DTE to DCE connection showing the minimum wiring required under the RS-232 interface standard.

require this signal before they will accept or transmit data. In applications where no modem is present, this pin is normally tied to pin 20, which in most cases is activated whenever the DTE is powered up.

Pin 22, Ring Indicator— The ring indicator signal is the means by which the DCE informs the DTE that the phone is ringing. All modems designed for direct connect to the phone network are equipped with auto answer. That is, the modem is able to recognize standard ringing voltage, indicate the ringing to the DTE, and answer (take the line off-the-hook) when told to do so by the DTE. The DTE tells the modem to answer the phone by activating pin 20, data terminal ready.

The 10 pins and signals we have just described to you are the ones most often used of those defined in the RS-232 standard.

Pin 15, Transmit Clock

Pin 17, Receiver Clock

Pin 21, Signal Quality Detector

Pin 24, External Clock— Synchronous modems use the signals on these pins. Pins 15, 17, and 24 control bit timing. Pin 21 indicates that the quality of the received carrier signal is satisfactory. Because the transmitting modem must send something (either a 0 or a 1) at each bit time, the modem controls the timing of the bits from the DTE. In turn, the receiving modem must output a bit and associated timing whenever received. Pin 15 (Transmitter Signal Element Timing—DCE

source), and pin 17 (Receiver Signal Element Timing—DTE source) are used for these purposes.

Pin 23, Data Rate Select— This entry according to figure 7-25 looks like there should be two pins assigned, but actually it is either data rate select (DTE source) or data rate select (DCE source). Some modems, called dual-rate modems, allow switching between two transmission speeds. Sometimes the speed is selected automatically by the modem during the initializing sequence, or it may be selected by the transmitting DTE. The signal on pin 23 determines whether the modem uses the low or high speed. Usually the modem at the calling end sets the speed for the connection and informs its DTE. The calling modem signals the speed to the answering modem, which informs the called DTE by activating data rate select (DCE source).

Pin 12, Secondary Data Carrier Detect

Pin 13, Secondary Clear to Send

Pin 14, Secondary Transmitted Data

Pin 16, Secondary Received Data

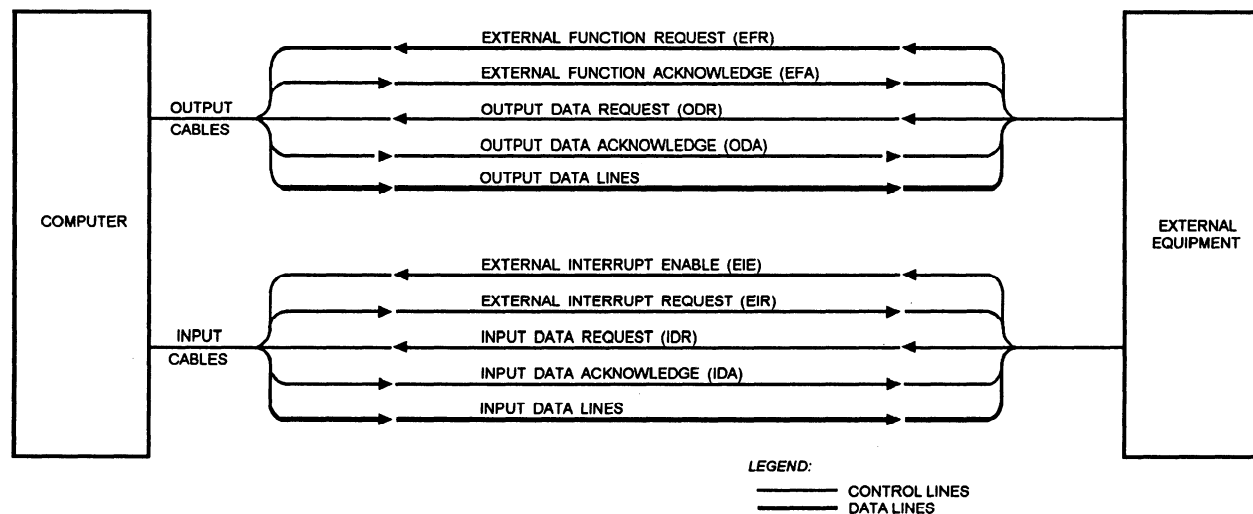
Pin 19, Secondary Request to Send— Some modems are equipped with both primary and secondary channels. The five secondary signals listed allow control of the secondary channel in the same way as described for the primary channel (pins 2, 3, 4, 5, and 8). In these modems, the primary transmission channel usually has the higher data rate, and the secondary channel transmits in the reverse direction with a much lower data rate, for example, 75 bps. Other signals that could be used (depending on the interface used) but not discussed include: send common, receive common, terminal in service, new signal, select

frequency, local loopback, remote loopback, test mode, select standby, and standby indicator.

I/O PARALLEL DATA OPERATIONS

Parallel data operations provide a multiwire communication path between the computer and one or more peripheral equipments. Parallel data operations use a **request/acknowledge** protocol. Generally speaking, a parallel channel is designed to transfer all the bits of a given byte or memory word, depending on the size of the computer and interface requirements, simultaneously. There is a separate data path or line for each bit that makes up the byte or word. The parallel channel handles data bytes or words in the same manner as the internal workings of the computer. There is no requirement to convert the byte or word to a sequential bit stream as there is in serial channel operations. There is, however, the need to **drive** or **receive (detect)** the digital signals over the I/O cables. The IOA or line driver/receiver on a pcb provides the means to accomplish this.

With parallel operations, there are two ways the computer can communicate with each external device. The computer can use a single cable to handle the parallel input and/or output operations or two cables: an **input** cable for the computer to receive information from an external device and an **output** cable for the computer to send information out to the same external device. The two cables will constitute one channel. Some computers can have up to 64 I/O channels. The I/O channels are usually identified by the octal numbering system. Thus, if you had a computer with 16 channels, the octal number assignments would be 0₈ through 17₈. Also, the channels are often arranged in groups with 4 channels per group. The parallel channel itself (fig. 7-27) consists of 8 or more data lines (8, 16,



ETFC0088

Figure 7-27.—Example of parallel channel architecture (two cables).

30, 32, 64, and so forth), and a number of control lines for passing signals that govern the transfer of information and coordinate operations of the computer and the peripheral device. In most computers, the data lines themselves are used to transmit control information (external functions) to the peripheral device, and to pass device status (status words and interrupt codes) to the computer.

We discuss parallel **computer to external devices** data operations using **one** and **two** cables; then we discuss intercomputer parallel data operations.

Computer to External Equipment (Single Cable)

With the computer to external equipment (single cable) set up, all the signals required to carry out parallel data operations are contained on a single cable. The number of lines in this setup can vary from 7 to 25; it will depend on the computer and the external device(s). We use an 8-bit computer as example of the lines used by a single parallel cable format (fig. 7-28). Other signals that could be used, but are not discussed, include: page end, auto feed, error, initialize external device (specific device name), and select input.

GROUND.— The ground signal ensures there is a complete circuit so there is current, thus enabling the signals to flow through the conductor and not collect at one end of the circuit (conductor). There are two grounds: one is a signal ground and the other a chassis ground connected to the device's chassis or ground. These signals do not move in either direction.

DATA STROBE.— The data strobe is sent from the computer to the external device. This signals the

external device that information is ready to be read from the data lines. The computer first puts the signals for all the data bits on the data lines, waits briefly to be sure the signal is stable, and then activates the data strobe line. When the external device sees that the data strobe signal has been sent, it accepts the character from the eight data lines.

BUSY SIGNAL.— The busy signal is sent from the external device to the computer to tell it not to send any more data. The external device maybe busy for various reasons. For example, it may still be in the process of obtaining information or the buffer maybe full.

SELECT SIGNAL.— The select signal usually corresponds to some sort of switch that must be in the enabled position by the external device. An example is an ONLINE switch on a printer. If it is disabled, the computer will be able to sense that something is wrong.

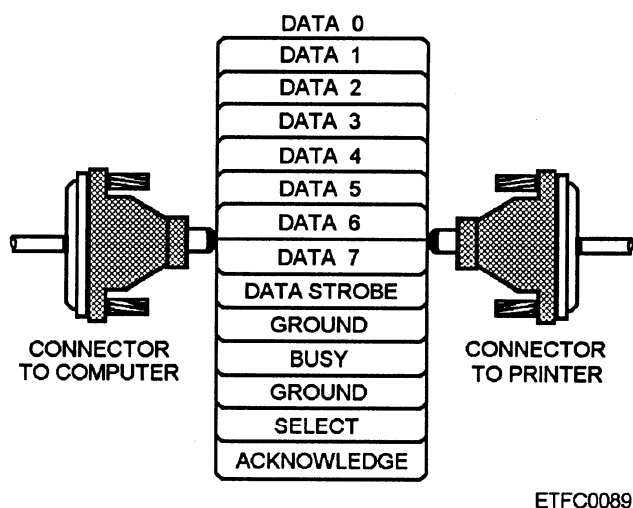
ACKNOWLEDGE SIGNAL.— The acknowledge signal is sent from the device to the computer to say that it has successfully received information (a character is this case). Thus instead of sending information at a constant rate, the computer waits for a positive indication that each character has been received before sending the next one.

DATA LINES.— Input/output data and interrupt bits are sent or received from the computer on these lines.

Single Cable Sequence of Events

The general sequence of events for a single cable parallel operations is as follows:

1. The computer puts the character on the data lines and sends the data strobe signal to tell the external device the data is there.
2. As soon as the external device sees the data strobe, it turns on the busy signal, telling the computer to wait while it reads the character from the data lines into its buffer.
3. Once the external device has processed the character, it sends the acknowledge signal and simultaneously removes the busy signal.
4. This tells the computer that it is all right to send another character and the process is repeated.



ETFC0089

Figure 7-28.—Single parallel cable.

Computer to External Equipment (Two Cables)

Two cables will make up one channel. As stated with two cables, one cable will specifically handle input functions and the other cable will handle **output** functions. Refer again to figure 7-27. Notice the direction of information flow. Data request signals are always sent from the external equipment to the computer. The acknowledge signals are always sent from the computer to the external equipment.

INPUT CABLE.— The input cable contains control lines and data lines. The number of lines will vary with the type of computer. They range from 8 to 64 lines. The operating mode (single, dual, and so on) has an effect on the number of lines affected. External devices send input data words and interrupt codes to the computer via input data (ID) lines. The information carried over these lines is as follows:

- External Interrupt Enable (EIE) —The computer sends the external interrupt enable signal to the external device to indicate it is ready to accept an external interrupt code word on that channel.

- Input Data Request (IDR) —The input data request control signal accompanies each input data word sent to the computer from the external device. The external device informs the computer that it has placed an input data word on the lines.

- External Interrupt Request (EIR) —The external interrupt request control signal accompanies each interrupt code sent to the computer from the external device. It informs the computer that an interrupt code is on the data lines.

- Input Data Acknowledge (IDA) —The input data acknowledge control signal informs the external equipment that the computer has sampled the input word or interrupt code on the input data lines on that channel.

OUTPUT CABLE.— The output cable contains control lines and data lines. Again the number of lines will vary with the type of computer. They range from 8 to 64 lines. The operating mode (single, dual, and so on) has an effect on the number of lines affected. Output data words and external function words are sent to the external device via data lines. The information carried over these lines is as follows:

- External Function Request (EFR) —The external device sends the external function request signal to the computer indicating that it is ready to accept an EF code word on that channel.

- External Function Acknowledge (EFA) —The computer sends the external function acknowledge signal to the external device indicating that it has placed an EF code word on the OD lines of that channel. This signal accompanies each function codeword sent to the external device.

- Output Data Request (ODR) —The external device sends the output data request control signal to the computer indicating that it is ready to accept an output data word.

- Output Data Acknowledge (ODA) —The computer sends the output data acknowledge signal to the external device indicating it has placed a word of data on the OD lines of that channel. This signal accompanies each output data word sent to the external device. It informs the external device that an output data word is on the data lines.

Two Cable Sequence of Events

The sequence of events using an input, output, external function (buffered), and external interrupt operations is described from the computer's point of view. We begin from the point that an input **data (ID)**, **output data (OD)**, **an external function (EF)**, or an **external interrupt (EI)** has been established for a channel. The computer and the external equipment on that channel transfer data as described in the following paragraphs. Refer back to figure 7-27.

INPUT DATA (ID) SEQUENCE OF EVENTS.— We begin from the point that an ID has been established for a channel. The computer and the external equipment on that channel will do the following to transfer data:

1. The external equipment places a word of data on the ID lines.
2. The external equipment sets the IDR line to indicate that a word of data is on the ID lines.
3. The computer detects the setting of the IDR line in accordance with internal priorities.
4. The computer samples the data word that is on the ID lines.

5. The computer sets the IDA line, indicating that it has sampled the data word on the ID lines.
6. The external equipment detects the setting of the IDA line. The external equipment may clear the IDR line anytime after detecting the setting of the IDA line, but will clear the IDR before the computer will recognize the next IDR.
7. The computer clears the IDA line before reading the next word on the ID lines.

OUTPUT DATA (OD) SEQUENCE OF EVENTS.— We begin from the point that an OD has been established for a channel. The computer and the external equipment on that channel will do the following to transfer data:

1. When the external equipment is ready to accept data, it sets the ODR line (this may already have happened before the OD buffer was established).
2. The computer detects the setting of the ODR line in accordance with internal priorities.
3. The computer places a word of data on the OD lines.
4. The computer sets the ODA line to indicate that a word of data is on the OD lines.
5. The external equipment detects the setting of the ODA lines. (The external equipment may clear the ODR line anytime after detecting the setting of the ODA, but clears the ODR line before the computer will recognize the next ODR).
6. The external equipment samples the data word that is on the OD lines.
7. The computer clears the ODA line before placing the next word on the OD lines.

EXTERNAL FUNCTION (EF) SEQUENCE OF EVENTS (NORMAL).— We begin from the point that an EF has been established for a channel. The computer and the external equipment on that channel will do the following to transfer:

1. When the external equipment is ready to accept an EF code word, it sets the EFR line (this may have already happened before the EF buffer was established).
2. The computer detects the setting of the EFR line in accordance with internal priorities.
3. The computer places an EF code word on the OD lines.

4. The computer sets the EFA line to indicate that the EF codeword is on the OD lines.
5. The external equipment detects the setting of the EFA line. The external equipment may clear the EFR line anytime after detecting the setting of the EFA line, but clears the EFR line before the computer will recognize the next EFR.
6. The external equipment samples the EF code word that is on the OD lines.
7. The computer clears the EFA line before placing the next word on the OD lines.

Forced external functions are the same as normal external functions except the computer does not require an external function ready signal from the external equipment, so the computer will not be delayed by steps 1 and 2.

EXTERNAL INTERRUPT (EI) SEQUENCE OF EVENTS.— The computer and the external equipment do the following to transfer an EI code word:

1. The computer, under program control, sets the EIE line when ready to accept an EI.
2. The external equipment detects the state of the EIE line.
3. When the status requires that the computer be interrupted, the external equipment places an EI code word on the ID lines.
4. The external equipment sets the EIR line to indicate that the EI codeword is on the ID lines.
5. The computer detects the setting of the EIR line in accordance with internal priorities.
6. The computer samples the EI codeword that is on the ID lines.
7. The computer clears the EIE line.
8. The computer sets the IDA line.
9. The external equipment detects step (8) or both steps (7) and (8). The external equipment may clear the EIR line anytime after detecting the setting of the IDA line, but clears EIR line before the computer will recognize the next EIR.
10. The computer clears the IDA line before sampling the next word on the ID lines.

NOTE: Not all computers have the EIE lines; consult your computer's technical manual.

The computer and external device repeat these sequences for each successive word of data until they have transferred the block of data words specified by the input buffer control words.

Intercomputer I/O Operations

Parallel channels are often used to communicate between two stand-alone computers. In this mode, the computers will appear as external devices to each other. One computer will be designated the transmitting (outputting) computer; the other computer will be designated the receiving (input) computer. A similarity exists between intercomputer channels and normal channels. The two cables are identical; in this mode all the signals remain the same except ODA and ODR, which become ready and resume respectively. Figure 7-29 illustrates the interface between two computers.

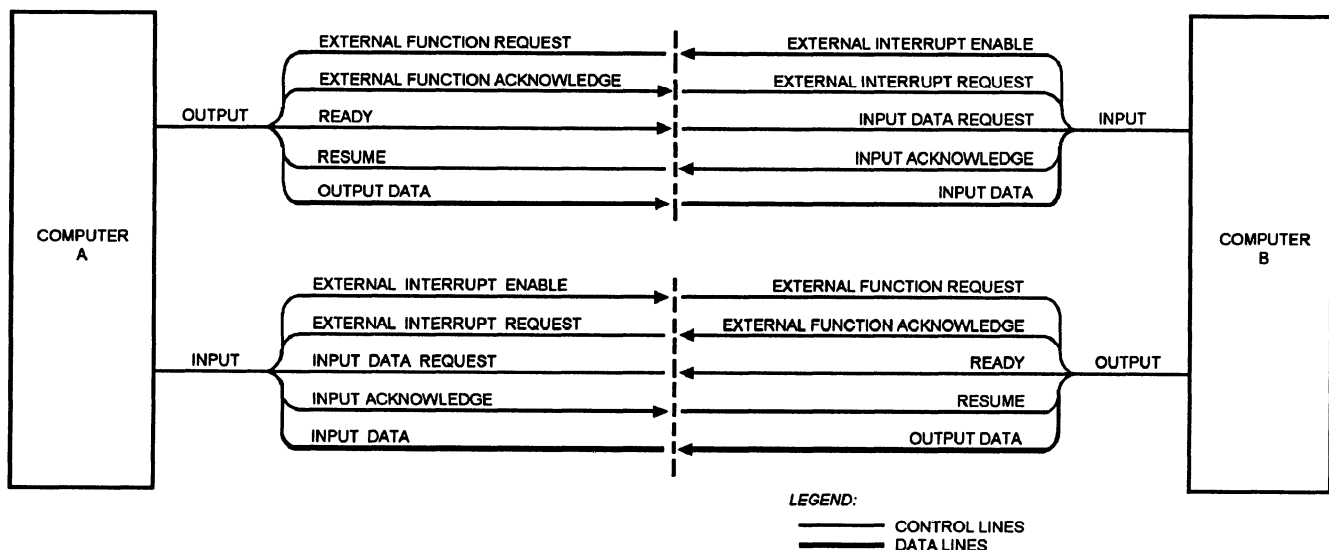
The two types of information transferred over the intercomputer channels data lines are **command words** and **data words**. Command words are used to exchange external function data, which includes external functions, forced external functions and external function buffer words, between the transmitting computer and the receiving computer. Data words are sent as part of output data buffers from the transmitting computer and accepted as part of the receiving computers input data buffer. Command words use additional interface signals to identify their function and to coordinate their transfer. When the transmitting computer generates an external function acknowledge signal with the ready signal, the data word transmitted is identified as a forced external function or an external function command word. The external

interrupt enable signal is set to identify the command word as an external function command word. If the external interrupt enable is not set, the command word is a forced external function.

The sequence of events for intercomputer command word and data transfers is as described in the following paragraphs.

INTERCOMPUTER COMMAND WORD TRANSFER (BUFFERED).— Whenever the transmitting computer has an EFR line and the receiving computer has an EIE line, transfer of **buffered** command words is possible. As you read, refer to figure 7-29; we designate computer A as the sending computer and computer B as the receiving computer. Whenever an EF buffer has been established in the transmitting computer for a channel, the transmitting computer and the receiving computer do the following to transfer a command word:

1. Computer B, under program control, sets the EIE line when it is ready to accept an EF command word from computer A.
2. In accordance with internal priority, computer A recognizes the EIE as an EFR and places the EF code on the data lines. The EF command word will be held on the data lines until computer B sets the resume line or until computer A's program intervenes to resolve the no resume condition.
3. Computer A sets the EFA line to indicate that the EF command word is on the OD lines.



ETFC0090

Figure 7-29.—Intercomputer interface.

4. In accordance with internal priorities, computer B detects the setting of the EFA line of computer A (which will be recognized as the EIR line) and samples the ID lines.
5. Computer B clears the EIE line.
6. Computer B sets the IDA line.
7. Computer A detects the setting of the IDA line of computer B (which will be recognized as the resume line).
8. Computer A clears the EFA line before placing the next word on the OD lines, and computer B clears the IDA line before reading the next word on the ID lines.

NOTE: Whenever the transmitting computer does not have an EFR line, or the receiving computer does not have an EIE line, a command will be transferred with force. For forced transfers, step 3 and step 7 are not used.

Computer A and computer B repeat this sequence for each successive command word until they have transferred the block of command words specified by computer B's EF buffer control words.

INTERCOMPUTER DATA TRANSFER.— Whenever an OD buffer has been established in computer A and an ID buffer has been established in computer B for the same channel, computer A and computer B transfer data. Again refer to figure 7-29 with computer A as the sending computer and computer B as the receiving computer. The sequence is performed as follows:

1. Computer A places a word of data on the OD lines. The OD word is held on the data lines until computer B sets the resume line, or until computer A's program intervenes to resolve the no resume condition.
2. Computer A sets the ready line to indicate that a word of data is on the OD lines.
3. In accordance with internal priorities, computer B detects the setting of the ready line of computer A (which will be recognized as the IDR line).
4. Computer B samples the ID lines.
5. Computer B sets the IDA line.

6. Computer A detects the setting of the IDA line of computer B (which will be recognized as the resume line).
7. Computer A clears the ready line before placing the next word of data on the OD lines, and computer B clears the IDA line before sampling the next word of data on the ID lines.

Computer A and computer B repeat this sequence until they have transferred the block of words specified by the buffer control words. Buffer lengths specified by each computer are the same.

SUMMARY—INPUT/OUTPUT (I/O) AND INTERFACING

This chapter has introduced you to how computers communicate with and control other computers and external devices. The following information summarizes important points you should have learned:

I/O ORGANIZATION— All computers are capable of I/O operations. Some rely on the CPU to handle I/O operations. Others have an I/O processor (IOC). An I/O processor enables the computer to perform other operations while still performing I/O operations.

I/O PROCESSOR— An I/O processor (IOC) controls the transfer of information between the computer's main memory and the external equipments. IOCs are packaged in (1) IOC/IOA modules or multiple IOC/IOA pcb's, and (2) I/O pcb's. The IOC relieves the CPU of the necessity to perform the time consuming functions of establishing, directing, and monitoring transfers with external equipments. Data and control signals are exchanged with external equipments via the IOA. The IOA changes the input and output control and data signal voltages to the voltage requirements of the computer or external equipments. Communication between the IOC and the IOA is by means of a bidirectional bus.

I/O DATA ARRANGEMENTS— The types of information exchanged between the computer and the external equipments fall into two basic categories: data words and control words. Data words represent the alphabetic and numeric information exchanged. Control words specify an action to be accomplished by an external equipment.

I/O DATA FORMATS— Computers exchange data in either parallel or serial format. When the computer uses a parallel configuration, all bits of information represented by a byte or word are input or

output simultaneously. When the computer uses a serial configuration, all bits of information are input or output one at a time.

I/O INSTRUCTIONS— All computers have I/O instructions. Command instructions are executed by the IOC under the control of the CPU's main program. They provide control over IOC single-and dual-channel operations. A chain consists of IOC control words, command words, output data words, and specified locations for external status words and data words returned (input) from the channel.

I/O OPERATIONS— Input/output operations are initiated by the CPU. Computers with an IOC begin I/O control functions only after an initiate I/O or equivalent instruction is executed by the CPU. Computer instructions inform the external equipment which type of operations to perform with function codes. They also specify memory areas for input and output information.

OPERATING MODES— I/O operations include both digital and linear ICs. The linear IC circuits are the first and last type of circuitry the information interfaces with when entering and leaving the computer. Registers in I/O operations provide the interfacing between the CPU, I/O, and memory. They enable and route control and data information between the CPU, I/O, and memory using the internal bus system. The data registers are used to hold or buffer data during interchanges between the very fast CPU and the slower external equipments. The status registers hold information for the CPU that indicates the operating condition and current activities of the external equipments.

I/O FUNCTIONS— The input and output functions performed by an I/O processor are defined and enabled through the interpretation and execution of input/output and/or input/output controller (I/O(C)) commands obtained from main memory.

DIRECT CPU INTERFACE— With direct communication, also called accumulator-based I/O, the peripheral devices are tied directly into the CPU communication bus (control bus, data bus, and so forth). In a simple I/O scheme, the CPU handles all I/O transactions by executing one or more instructions for each word of information transferred.

DIRECT MEMORY ACCESS (DMA)— DMA allows blocks of information to be transferred directly in and out of memory and from and to an external device without any CPU intervention. Information is transferred at a speed compatible with that of the

external device. A DMA controller is usually placed between the external device and the computer's bus.

I/O INTERFACING— Computers may have a small number of channels or ports with multiple equipments connected to each channel; or they may, particularly in larger computers, have a number of I/O channels with limited numbers or types of external equipments on each channel or port.

I/O INTERFACING STANDARDS— There are two major types of computer/external equipment communication formats: serial and parallel. The communication formats are governed by the standard that is identified by the interface. As a general rule, the standards can be divided into four categories: mechanical, electrical, functional, and procedural.

I/O INTERFACING COMPONENTS— The computer's I/O processor must ensure that the voltage levels between the computer and the external equipments are compatible. The primary circuitry that accomplishes this is located on an I/O pcb or modules/pcb's that make up an IOA. Some of the primary I/O interfacing hardware include universal receiver-transmitters, line drivers, and line receivers.

UNIVERSAL RECEIVER-TRANSMITTER— Within a digital computer, the data is transferred internally using a parallel format. All the bits of a byte or memory word are exchanged simultaneously between registers, buses, and other computer logic. For the data to be communicated over a serial channel, it must be converted from parallel to a serial bit stream. The USART is designed to function as a peripheral device to the microprocessor. The actual conversion from serial to parallel or parallel to serial is performed by the USART and is transparent to the microprocessor. The standard USART chip is comprised of logic circuits, which are connected by an internal data bus.

LINE DRIVERS/RECEIVERS— The line drivers/receivers are designed to send and receive signals over short or long distances using serial or parallel format. Large voltages or currents are generated from small voltage or current using TTL or MOS circuitry. The two types most commonly used are single-ended and differential.

I/O INTERFACE FORMATS— There is a variety of serial and parallel I/O channel formats. Your computer's technical manual will provide the standards to be used with the cabinet and cable connectors. They will match the standards that govern the requirements for parallel and serial interfacing.

I/O SERIAL DATA OPERATIONS— Serial data operations exchange information via a single path, line, or wire. The channel/port itself is made up of several wires, but only one is used to transfer the binary data.

INTERCOMPUTER I/O OPERATIONS— Parallel channels are often used to communicate between two stand-alone computers. In this mode, the computers will appear as external devices to each other. One computer will be designated the transmitting

(outputting) computer; the other computer will be designated the receiving (input) computer.

Learn all you can about how input/output operations enable the computer to communicate with and control the variety of equipments used in today's computer systems. Learn about the internal I/O process and the interfacing process. This will help you to troubleshoot and diagnose input/output problems and to repair and/or replace I/O parts.